ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

ANÁLISIS DE METODOLOGÍAS PARA DESARROLLO DE AGENTES, CASO PRÁCTICO: BUSCADOR INTELIGENTE PARA UNA INTRANET

Tesis previo a la obtención del Grado de

MASTER EN INFORMATICA EDUCATIVA

AUTOR: Carrillo Medina José Luis

TUTOR: Dr. Hugo Banda MsC.

Riobamba – Ecuador

2005

RESUMEN

El ser humano a lo largo de su historia ha buscado siempre medios o mecanismos para (acuitar e] desarrollo de sus tareas. Kn la era industrial, por ejemplo, el ser humano valido de su ingenio creó instrumentos que fueran la extensión de las capacidades típicas de su cuerpo. Hoy an día, en la que se puede llamar la hra del Conocimiento, el ser humano, a través de la tecnología de la información, busca extender las capacidades de la mente humana, En esta búsqueda se han definido herramienlas. que tratan de expandir el conocimiento, enlrc ellas se puede citar: las herramientas Multimedia, llypermedía y la Inteligencia Artificial. Dentro de este último enfoque surge como paradigma interesante el de Agentes de Software. Un agente de Software, según Marvin Minsky, es "un programa computa!; ion al cnn cierta inteligencia" [MINSKY94]: u coma lo propone [JANCAQ5] "una entidad sotlware a la que se pueden delegar tareas", relacionadas con proceses mentales que únicamenle que el ser humano puede realizar. Varios esfucr/os en la Ingeniería de Software desde sus eomien/os, se han centrado en la construcción de abstracciones lo suficientemente poderosas y naturales para modelar, disertar y desarrollar sistemas complejos. Se puede citar como ejemplos: las abstracciones procedí menta les. los tipo de dalos abstractos y, más recientemente, los objetos. Algunos autores, como [WOOI.D95a: MAES91J creen que los agentes representan un avance similar. Según ellos, y debido a las tendencias recientes, la nalurale/a dinámica y distribuida del dato, asi como de las aplicaciones, requiere que el software no sólo responda a pedidos de sus usuarios sino que además anticipe de forma inteligente, que pueda adaptarse y busque activamente caminos para asistir a los usuarios, tn respuesta a estos requerimientos varios investigadores coinciden, que un camino particularmente interesante parece ser el desarrollo del paradigma: Agente, lo que constituye uno de los aspecto fundamentales del presente.

SUMMARY

Human beings throughout history has always sought means or mechanisms (acuitar and] carry out their tasks. Kn industrial era, for example, used its human ingenuity created instruments that were the extent of typical capabilities your body. Today an day, which you can call the HRA of Knowledge, man, through information technology, looking to extend the capabilities of the human mind, in this search are defined herramienlas. that try to expand knowledge, enlrc them can be mentioned:.. the Multimedia tools, llypermedía and Artificial Intelligence Within the latter approach comes as interesting paradigm of software agents software agent, as Marvin Minsky, is "a program !; computed ion the cnn some intelligence "[MINSKY94]: u eat proposes [JANCAQ5] a" sotlware entity that can delegate tasks "related to mental proceses that únicamenle that humans can perform. Several esfucr / OS software engineering since its eomien / os, have focused on building powerful enough abstractions and natural modeling, lecturing and developing complex systems. One can cite as examples: abstractions proceeded mint them. dalos the abstract type and, more recently objects. Some authors, such as [WOOI.D95a: MAES91J believe that agents represent a similar breakthrough. According to them, due to recent trends, nalurale / a dynamic and distributed the data, as well as applications, requires that the software does not only respond to requests from its users but also intelligently anticipates that can adapt and actively seek ways to assist users, tn response to these requirements several researchers agree that a particularly interesting way appears to be the development paradigm: Agent, which is a fundamental aspect of this.

INTRODUCCIÓN

Con mucho acierto se proclama que la humanidad ha iniciado la Era del Conocimiento, en razón de que el hombre no solamente utiliza en todos los campos de su actuación la información que dispone, sino también busca permanentemente aplicaciones a las situaciones mas diversas de la vida diaria. De esta visión del hombre actual y sobre todo en su proyección futura, se origina el paradigma Agente dentro del área de la Inteligencia Artificial.

Un indicativo de que la tecnología agente ha madurado, dentro de una parte significativa de la ciencia de la computación, es el gran número y variedad de aplicaciones que han aparecido recientemente [JENNIN96, ARENS96, HAYES95, ISHIZA96, LASHKA94, ETZION94]. Estas aplicaciones abarcan, entre otros, la interacción humano-computador, la ingeniería, el comercio, la recolección de datos y acceso a la información.

Como consecuencia de lo señalado, la industria ha comenzado a interesarse en adoptar esta tecnología para desarrollar sus propios productos. La introducción de la tecnología de agentes a la industria requiere de metodologías que asistan en todas las fases del ciclo de vida del sistema agente. Sin técnicas adecuadas para soportar este proceso, tales sistemas probablemente no serán lo suficientemente confiables, mantenibles o extensibles. Serán difíciles de comprender y sus elementos serán muy poco reusables.

Actualmente se está realizando un notable esfuerzo en el estudio de teorías, arquitecturas y lenguajes de agentes, así como en la definición de un conjunto de primitivas y procesos para normalizar las interacciones, pero todavía no se han extendido estos avances al plano de la metodología. Es por ello que este trabajo se centra en el estudio de posibles soportes conceptuales (modelos, metodologías y técnicas) de desarrollo de sistemas basados en agentes que podrían constituir un marco organizativo adecuado en la praxis del proceso de construcción de los mismos. Se realiza un análisis del valor que pueden tener metodologías convencionales (no orientada a agentes) y metodologías propuestas por la literatura especializada en el área de agentes.

CAPITULO 1
MARCO REFERENCIAL

1.1 TEMA

Análisis de Metodologías para desarrollo de Agentes. **CASO PRÁCTICO:** Buscador Inteligente para una Intranet.

1.2 PLANTEAMIENTO DEL PROBLEMA

El ser humano a lo largo de su historia ha buscado siempre mecanismos o medios para facilitar el desarrollo de sus actividades. En la era industrial, por ejemplo, el ser humano valido de su ingenio creó instrumentos que fueran la extensión de las capacidades de su cuerpo. Hoy en día, en que se puede llamar la Era del Conocimiento, el ser humano a través de la tecnología de la información busca extender las capacidades de su mente. En esta búsqueda se han definido herramientas que tratan de expandir el conocimiento, entre ellas se pueden citar: las herramientas Multimedia, Hypermedia y la Inteligencia Artificial. Dentro de este último enfoque surge como paradigma interesante el de Agentes de Software [PETS96].

El advenimiento de los agentes software dio inicio a una gran discusión sobre lo que constituye un agente y la manera en que éstos difieren de los programas en general. Los agentes software son probablemente el área de la tecnología de la información que crece de manera más rápida.

Un agente software, según Marvin Minsky, es “un programa computacional con cierta inteligencia” [MINSKY94]; o como lo propone [JANCA95] "una entidad software a la que se pueden delegar tareas", relacionadas con procesos mentales que únicamente el ser humano podría realizarlos.

La tecnología de agentes ha recibido gran atención en los últimos años y, como resultado, la industria está comenzando a interesarse en esta tecnología para desarrollar sus propios productos de agentes de software. Grandes empresas como IBM, Microsoft, Mitsubishi o BT cuentan ya con laboratorios de agentes inteligentes (Agentes de Software) y han sacado los primeros productos al mercado, principalmente de agentes de usuario e Internet. Sin embargo, para que el paradigma de la programación orientada a agentes se extienda hacia aplicaciones genéricas, es necesario que se desarrollen técnicas de análisis y diseño con este paradigma.
La investigación en metodologías orientadas a agentes es un campo incipiente, por lo que es necesario realizar un análisis sobre las metodologías para el desarrollo de agentes de software, las metodologías

orientadas a agentes (MOA) no han surgido como metodologías totalmente nuevas, sino que se han planteado como extensiones tanto a metodologías orientadas a objetos (MOO) como a metodologías de ingeniería del conocimiento (ICO). Debido a la relación del paradigma de la orientación a agentes con la orientación a objetos y con los sistemas basados en conocimiento

Como consecuencia, la introducción de la tecnología de agentes en la industria requiere de metodologías que asistan en todas las fases del ciclo de vida del sistema agente. Sin técnicas adecuadas para soportar este proceso, tales sistemas probablemente no serán lo suficientemente confiables, mantenibles o extensibles, serán difíciles de comprender y sus elementos no serán fácilmente reusables.

1.3 OBJETIVOS

1.3.1 General

Analizar las Metodologías Orientadas a Objetos y las Orientadas a Agentes, para determinar la más aplicable al diseño e implementación de agentes en nuestro medio.

1.3.2 Específicos

- Revisión del estado del arte de la tecnología de Agentes (definición, arquitecturas y metodologías).
- Evaluación e identificación de similitudes y diferencias entre los Paradigmas Orientado a Objeto y Orientado a Agente.
- Discusión de eventuales propuestas acerca del uso de una Metodología Orientada a Objetos y la Metodología Orientada a Agentes para el diseño e implementación de Agentes.
- Diseño e implementación de un prototipo de agente, aplicado a un Buscador Inteligente.

1.4 JUSTIFICACIÓN

La tecnología de agentes está recibiendo una gran atención en los últimos años y, como consecuencia la industria está comenzando a interesarse en adoptar esta tecnología para desarrollar sus propios productos. Sin embargo, a pesar del rápido desarrollo de teorías, arquitecturas y lenguajes de agentes, se ha realizado muy poco trabajo en la especificación (y aplicación) de técnicas para desarrollar aplicaciones empleando tecnología de agentes.

La introducción de la tecnología de agentes en la industria requiere de metodologías que asistan en todas las fases del ciclo de vida del Sistema Agentes. La necesidad del empleo de estas técnicas será más importante a medida que el número de agentes de los sistemas aumente. Aunque en la comunidad multiagente se ha mencionado que es necesario el desarrollo y aplicación de metodologías, sorprendentemente, se ha realizado muy poco esfuerzo investigador en esta área. Actualmente se está realizando un notable esfuerzo en el estudio de teoría, arquitecturas y lenguajes de agentes, así como en la definición de un conjunto de primitivas para normalizar las interacciones, pero todavía no se han extendido estos avances al plano de la metodología o ingeniería del conocimiento. Es por ello que la presente tesis se centra en el estudio de posibles soportes conceptuales (modelos, metodologías y técnicas) de desarrollo de sistemas basados en agentes que podrían constituir un marco organizativo adecuado en la praxis del proceso de construcción de los mismos. Para el efecto se realizará un análisis del valor que pueden tener metodologías convencionales (no orientada a agentes) y metodologías propuestas por la literatura especializada en el área de agentes, para ser plasmada en una aplicación práctica.

CAPITULO 2
MARCO TEÓRICO

En el marco teórico se expone teorías, enfoques teóricos, investigaciones relacionadas con las metodologías que se consideran validas para la aplicación del desarrollo de agentes software.
La primera parte de este capitulo introduce los conceptos relacionados con la orientación a objetos, posteriormente se presentan la definición y la arquitectura de un agente de software, a fin de formar una base teórica que permita tener una visión más clara del concepto agente, de modo que se pueda discutir finalmente las similitudes y diferencias entre ambos paradigmas (objetos y agentes).
En la segunda parte se explica sobre las metodologías orientadas a objetos que se consideran las mas aplicables en el campo de desarrollo de un sistema agente.
En la tercera parte se explica sobre las metodologías orientadas a agentes que son extensiones de las metodologías orientadas a objetos y extensiones a las metodologías basadas en la ingeniería del conocimiento, para el desarrollo de un sistema agente.

1.5 OBJETOS Y AGENTES

1.5.1 Objetos: Fundamentos

En esta sección se esboza sintéticamente los principales fundamentos de la Orientación a Objetos de modo que se pueda realizar. en las secciones posteriores, un análisis de la relación entre agentes y objetos, en cuanto a similitudes y diferencias. Debido al objetivo de este trabajo, no se pretende realizar un análisis exhaustivo del paradigma Orientado a Objetos.

Cuando se utiliza métodos orientados a objetos para analizar o diseñar sistemas software complejos, las unidades básicas de construcción son clases y objetos.
Según Booch: "U*n objeto representa una unidad o entidad individual, tanto real o abstracta, con un rol bien definido dentro del dominio de aplicación... Un objeto tiene estado, comportamiento e identidad... Una clase es un conjunto de objetos que comparten una estructura común y un comportamiento común"* [BOOCH94].

El estado de un objeto abarca todas las propiedades del objeto más los valores actuales de cada una de estas propiedades. El comportamiento de un objeto es una función de su estado así como de las operaciones efectuadas sobre el mismo. Mientras que la identidad del objeto es la propiedad que lo distingue del resto de los objetos. Cada objeto creado durante la ejecución de un sistema orientado a objetos tiene una

identidad única, independientemente de los valores de los atributos del objeto. En particular, dos objetos con diferentes identidades pueden tener estados idénticos, y además, el estado de un objeto puede cambiar en tiempo de ejecución, sin que esto afecte a la identidad del objeto. En otras palabras la identidad implica que un objeto tiene una existencia independiente de su contenido [MEYER97].

Mientras que un objeto es una entidad concreta que existe en el tiempo y en el espacio, una clase representa sólo una abstracción, la "esencia" de un objeto, que captura la estructura y el comportamiento común a objetos relacionados.

En una clase se puede identificar dos componentes: la interfaz de la clase y su implementación [MEYER97].
La interfaz de una clase provee la vista externa y por tanto enfatiza la abstracción mientras que oculta la estructura y los secretos del comportamiento. En contraste, la implementación representa la vista interna de una clase encerrando los secretos de comportamiento.

Los objetos y las clases no existen en forma aislada. Se puede establecer relaciones entre clases por una de dos razones. Primero, una relación entre clases puede indicar algún tipo de compartición como ser estructuras o comportamiento común. Segundo, una relación entre clases puede significar algún tipo de conexión semántica.

Como lo establece Booch, existen tres tipos básicos de relaciones entre clases:

- La primera de estas es la generalización/especialización, que denota una relación "es un".
- La segunda es la relación todo/parte, y
- La tercera es la asociación, que denota algún tipo de dependencia semántica.

Uno de los conceptos centrales de la Orientación a Objetos es la herencia. La herencia es una relación de generalización/especialización por la cual, una clase comparte la estructura y/o comportamiento definidos en una o más clases. La clase de la cual se hereda, se llama superclase, mientras que la clase que hereda se llama subclase. Una subclase típicamente aumenta o restringe la estructura y el comportamiento de su superclase.

1.5.2 Agentes: Definición

Los agentes software son probablemente el área de la tecnología de la información que crece de manera más rápida.

Existen varias definiciones sobre agentes de software en las cuales se puede identificar similitud de conceptos entre ellas, por cuanto enfatizan sobre las mismas características; sin embargo, algunas agregan nuevas características, como se observa en las definiciones que se señala a continuación.

Definición de Genesereth y otros: “Una entidad es un agente software si y sólo si se comunica correctamente en un lenguaje de comunicación de agentes” [GENESE95]. Aquí el punto central (y único) es la comunicación del agente. Pero no está muy claro si la comunicación es entre agentes, entre un agente y un programa cualquiera, o entre un agente y un usuario humano.

Una definición relacionada con la anterior, que detalla la forma de comunicación, es la propuesta por Coen (agente SodaBot): “los agentes son programas que se comprometen en diálogo, negocian y coordinan transferencia de información”. Sin embargo, a pesar del detalle, aún falta la definición de los interlocutores.
[http://www.ai.mit.edu/people/sodabot/slideshow/total/P001.html]

En la definición de Virdhagriswaran se introduce un componente muy importante, “***la inteligencia***” (no estudiada en las definiciones anteriores) y que debería aparecer en todas las definiciones de agente. Según Virdhagriswaran: “el término agente es utilizado para representar dos conceptos ortogonales. El primero es la habilidad del agente para la ejecución simbólica. El segundo es la habilidad del agente para realizar razonamiento orientado hacia el dominio”.
[http://www.crystaliz.com/logicware/mubot.html].

Según IBM: “los agentes inteligentes son entidades software que realizan algún conjunto de operaciones en beneficio de un usuario o de otro programa con cierto grado de independencia o autonomía, y realizando esto, emplean cierto conocimiento o representación de los deseos y objetivos del usuario”.
[http://activist.gpl.ibm.com:81/WhitePaper/ptc2.htm]

Esta definición, ve a un agente inteligente como una entidad que actúa en favor de otros, con autoridad otorgada por los otros. Una característica interesante, que introduce esta definición, es la necesidad de mantener cierto conocimiento referente al entorno y al usuario.

La definición de Brustoloni, [BRUSTO91], según la cual: "los agentes autónomos son sistemas capaces de acciones intencionadas y autónomas en el mundo real", comparte con la definición propuesta por IBM la autonomía, pero el ambiente en el cual actúa el agente no está muy bien definido, pudiendo dar lugar a múltiples interpretaciones.

Las siguientes definiciones comparten la necesidad de sensores y efectores en el agente, de modo que a través del primero se pueda monitorear el ambiente y a través del último se actúe sobre el ambiente:

- *Definición de Rusell y Norving (agente AIMA):* "Un agente es cualquier cosa que puede ser vista como algo que percibe su ambiente a través de sensores y actúa sobre su ambiente a través de efectores" [RUSELL95].

- *Definición de Maes:* "Los Agentes autónomos son sistemas computacionales que habitan en algún ambiente dinámico complejo, monitorean y actúan de forma autónoma en este ambiente, y realizando esto logran un conjunto de objetivos o tareas para los cuales están diseñados"[MAES95]. En esta definición el ambiente tiene características de dinamismo y complejidad (no especificadas en la definición de Rusell y Norvig), el agente actúa en forma autónoma y mantiene una serie de propósitos (tareas y objetivos).

- *Definición de Hayes-Roth*: "Los agentes inteligentes continuamente realizan tres funciones: percepción de condiciones dinámicas en el ambiente; acción para afectar condiciones en el ambiente; y razonamiento para interpretar las percepciones, resolver problemas, realizar inferencias y determinar acciones" [HAYES95]. En esta definición el comportamiento del agente para lograr acciones se presenta con mayor detalle, como una serie de pasos a seguir: monitorear eventos, razonar sobre los mismos, inferir posibles acciones y finalmente ejecución de la acción.

Por último, la definición de Smith et al (el agente KidSim), introduce un nuevo componente: la persistencia. "Definamos un agente como una entidad software persistente dedicada a un

propósito específico. ‘Persistente’ distingue a un agente de las subrutinas; los agentes tienen sus propias ideas acerca de cómo cumplir las tareas, su propia agenda. ‘Propósito específico’ los distingue de todas las aplicaciones multifunción; los agentes son típicamente más pequeños” [SMITH94].

Todas estas definiciones tratan de dar una noción del concepto Agente. En la mayoría de ellas está presente la autonomía como característica principal, y el hecho de que los agentes habitan un determinado ambiente (algunas de las definiciones citadas restringen este ambiente mientras que otras dejan abierta la cuestión y por tanto cualquiera puede interpretarlo como mejor le parezca). Algunas de las definiciones introducen la necesidad de comunicación de los agentes con otros agentes y también con el usuario humano. En la definición del agente KidSim se introduce además algo que se considera importante, el concepto de persistencia.

En un trabajo realizado por Stan Franklin y Art Graesser, [GRAESS96], se realiza un análisis de distintas definiciones y se concluye a partir de ellas, usos comunes de la palabra Agente. La conclusión es la siguiente: *Las definiciones... parecen derivar en uno o ambos usos comunes de la palabra agente:*

1) *algo que actúa o que puede actuar*
2) *algo que actúa en lugar de otro con su autorización*

El punto 1), lo verifica cualquier programa de computadora, si se entiende por actuar el simple hecho de ejecutar cierta tarea. El punto 2) aparentemente sufre nuevamente de la falta de una definición clara del término ‘actuar’. Si ‘actuar’ significara, iniciar una acción movido por sus propios intereses u objetivos, varios programas de computadoras serían eliminados de esta categoría y se estaría frente a una clasificación más fina. Pero también podría ser considerado como ejecutar simples comandos en respuesta a pedidos del usuario. Sin embargo en el trabajo que ellos presentan no existe una definición de ‘actuar’ y por tanto no se puede deducir conclusiones.

Posiblemente, una mejor ayuda a la comprensión del concepto de Agente la ofrecen Franklin y Graesser, al presentar una lista de cualidades que compartirían los agentes y por tanto constituirían la esencia de lo que es un agente:

- *Cada uno está situado en y es parte de algún ambiente.*
- *Cada uno monitorea su ambiente y actúa de forma autónoma sobre él.*
- *No se necesita ninguna otra entidad para alimentarlo con entradas o para interpretar y utilizar sus salidas.*
- *Cada uno actúa de tal manera que su acción actual puede afectar su monitoreo posterior, esto es sus acciones afectan a su ambiente.*
- *Cada uno actúa en prosecución de su propia agenda.*
- *Cada uno actúa contínuamente durante cierto periodo de tiempo.*

En esta lista el término 'actuar', anteriormente ambiguo, modificado ahora con el adjetivo autónomo nos permite considerar que 'actuar' significa iniciar una acción direccionada por sus propios intereses u objetivos y podría ser válida.

Luego de la revisión de varias de las definiciones existentes, a criterio del autor, la que puede ser considerada como la más clara y completa es la propuesta por Wooldridge y Jennings [WOOLD95a]. ***Esta definición es la que se adoptará como base para todo el análisis posterior***.

- *Definición: "Un sistema computacional hardware o software caracterizado por las siguientes propiedades:*

    - *autonomía: los agentes operan sin una directa intervención de humanos u otros, y tienen cierto grado de control sobre sus acciones y su estado interno;*
    - *habilidad social: los agentes interactúan con otros agentes (y posiblemente con humanos) vía algún tipo de lenguaje de comunicación entre agentes;*
    - *reactividad: los agentes perciben su ambiente, (que puede ser el mundo físico, un usuario vía una interfaz gráfica, una colección de otros agentes, la INTERNET, o tal vez todos estos combinados), y responden de una manera 'timely' a cambios que ocurren en él;*
    - *pro-actividad: los agentes no actúan simplemente en respuesta a su ambiente, son capaces de exhibir comportamiento oportuno, dirigido por objetivos, tomando iniciativas cuando sea apropiado.*
    - *nociones mentales: un agente tiene creencias, deseos e intenciones.*

- *racionalidad: un agente realiza acciones a fin de lograr objetivos.*
- *veracidad: un agente no es capaz de comunicar información falsa a propósito.*
- *adaptabilidad o aprendizaje"*

1.5.3 Arquitectura de Agentes

Se presenta una breve descripción de varias de las arquitecturas de agentes desarrolladas hasta el momento.

Arquitectura de un sistema, según la ingeniería del software:

> " *... definimos un diseño del software como una descomposición del sistema en módulos – descripción de qué hace cada módulo y la relación entre estos módulos. Tal descripción es a menudo llamada la arquitectura del software o la estructura del software"* [GHEZZI91]

Por tanto la arquitectura de un sistema, comprende:

- división del sistema en módulos y descripción de cada uno de ellos, y
- relación de los distintos módulos componentes entre sí.

¿Qué es una arquitectura de agentes?

> ..."Una metodología particular para construir agentes. Especifica cómo... el agente puede ser descompuesto en un conjunto de módulos componentes y cómo estos módulos pueden interactuar. El conjunto total de módulos y sus interacciones deben proveer una respuesta a la pregunta de cómo el dato monitoreado y el estado interno del agente determinan las *acciones... y estados internos futuros."* [MAES91]

Esta definición está absolutamente de acuerdo con la definición de arquitectura propuesta por Ghezzi, y de hecho debe ser así puesto que un agente es un sistema software que posee cierta estructura.

En [WOOLD95, b], se propone una clasificación de arquitectura de agentes, según lo que considera como motor de acción del agente. Observa tres categorías principales: arquitecturas deliberativas, arquitecturas reactivas y arquitecturas híbridas.

Si bien, parece interesante el trabajo de Wooldridge, se considera que las denominaciones: arquitecturas deliberativas, arquitecturas reactivas y arquitecturas híbridas, no son adecuadas, ya que una arquitectura en sí es un modelo abstracto de un sistema y por tanto no posee ningún tipo de comportamiento, lo que realmente posee comportamiento es el sistema, es decir el agente que está siendo modelado. Es por eso que se cree conveniente y adecuado adoptar las denominaciones: arquitecturas para agentes deliberativos, arquitecturas para agentes reactivos y arquitecturas para agentes híbridos.

1.5.3.1 Arquitecturas para Agentes Deliberativos

En [WOOLD95a] se define a una arquitectura de agente deliberativo como aquella que contiene un mundo representado explícitamente y un modelo lógico del mismo, y en la cual las decisiones (por ejemplo acerca de las acciones a realizar) son hechas por medio de un razonamiento lógico (o por lo menos pseudo-lógico), basado en concordancia de patrones y manipulación simbólica por ejemplo:

1.5.3.1.1 Planning Agents:

Desde inicios de los setenta, la comunidad de la Inteligencia Artificial dedicada al Planning (que es esencialmente programación automática; es decir, diseño de un curso de acción que, al ser ejecutado, resulta en el logro de algún objetivo deseado) ha estado fuertemente relacionado con el diseño de agentes. Parece razonable entonces, que muchas de las innovaciones en el diseño de agentes provengan de esa comunidad.

Tal vez el sistema mejor conocido basado en planning es STRIPS [FIKES71]. La arquitectura de este sistema contiene una descripción simbólica del mundo y el estado deseado de objetivos y del conjunto de descripciones de las acciones que caracterizan las pre y post condiciones asociadas con varias acciones. Tomando como base este conocimiento intenta encontrar una secuencia de acciones que lograrán el objetivo utilizando un análisis simple. El algoritmo de planeación de STRIP es muy sencillo y se probó que es ineficiente en problemas de complejidad moderada.

Se han realizado varios intentos para construir agentes cuyo componente principal sea un planeador. Por ejemplo: el sistema Integrated Planning, Execution and Monitoring (IPEM) que está basado en un planeador sofisticado no lineal [AMBORS88]; el sistema AUTODRIVE que tiene planning agents operando en un ambiente altamente dinámico [WOOD93]; el sistema PHOENIX que incluye agentes basados en planeadores [COHEN89].

1.5.3.1.2 Intellince Resource-bounded Machine Architecture *IRMA*

[BRATMA88]. Esta arquitectura tiene cuatro estructuras claves de datos simbólicos: una biblioteca de planes, una representación explícita de creencias, deseos e intenciones. Adicionalmente, la arquitectura tiene: un módulo para razonamiento acerca del mundo; un analizador para determinar qué planes deben ser utilizados para lograr las intenciones del agente; un filtrador de procesos; y un proceso de deliberación. El proceso de filtración es responsable de determinar el subconjunto de cursos de acción potenciales del agente, que tienen la propiedad de ser consistentes con las intenciones actuales del agente. La elección entre opciones que compiten lo realiza el proceso de deliberación.

1.5.3.1.3 HOMER

En [VERE90] se argumenta que el establecimiento de tecnologías para agentes inteligentes está lo suficientemente desarrollada, como para construir un agente prototipo autónomo con habilidades lingüísticas, capacidades de planeación y acción, etc. Se desarrolló tal agente y se lo llamó HOMER. La arquitectura de HOMER es específica para el problema de la

comunicación en lenguaje natural (reducido) con el usuario; para ello incluye una base de vocabulario limitada a un subconjunto del idioma Inglés con cerca de 800 palabras y una memoria episódica también limitada. HOMER toma instrucciones del usuario que pueden contener referencias temporales moderadamente sofisticadas; puede planear cómo realizar sus instrucciones, y puede luego ejecutar sus planes, modificándolos como sea requerido durante la ejecución. El agente además, gracias a su memoria episódica, es capaz de responder a preguntas acerca de sus experiencias pasadas.

1.5.3.1.4 GRATE

GRATE es una arquitectura en capas en la que el comportamiento de un agente es guiado por actitudes mentales tales como creencias, deseos, intenciones e intenciones colectivas [JENNIN93]. Los agentes se dividen en dos partes distintas: un sistema de nivel de dominio y una capa de cooperación y control. El primero resuelve problemas para la organización sea esta en el dominio de controles industriales, de finanzas o transporte. El último es un controlador de meta nivel que opera en el nivel de dominio del sistema con la intención de asegurar que las actividades del agente a nivel del dominio sean coordinadas con aquellas otras dentro de la comunidad. La capa de cooperación está compuesta de tres módulos genéricos: un módulo de control que es la interfaz con la capa de dominio del sistema, un módulo de fijación de situaciones y un módulo de cooperación. Los módulos de fijación y cooperación proveen una implementación de un módulo de responsabilidad colectiva, que especifica cómo los agentes deberían actuar tanto localmente como hacia otros agentes comprometidos a cooperar en la resolución del problema.

1.5.3.1.5 BDI

BDI significa creencias (Beliefs), deseos (Desires) e intenciones (Intentions), que son componentes mentales presentes en muchas arquitecturas de agentes. Las creencias representan el conocimiento del agente, los deseos representan los objetivos y las intenciones otorgan deliberación al agente. La exacta definición de estos términos varía de autor a autor.

Este tipo de arquitectura ve al sistema como un agente racional que tiene ciertas actitudes mentales, tales como: creencias, deseos e intenciones, representando respectivamente los estados de información, motivacional y deliberativos del agente. Estas actitudes mentales determinan el comportamiento del sistema y son críticos para lograr el desempeño adecuado u óptimo cuando la deliberación está sujeta a recursos limitados. [GEORGE95].

Las creencias de un agente representan el conocimiento del agente. El contenido del conocimiento puede ser cualquiera, por ejemplo conocimiento acerca del ambiente del agente o acerca de su historia. Los deseos son un conjunto de objetivos a largo plazo. Un objetivo es típicamente una descripción de un estado deseado del ambiente. Los deseos proveen al agente de la motivación para actuar. Los objetivos constituyentes de los deseos pueden ser contradictorios, entonces el sistema tiene que poder, de cierta forma, elegir qué objetivo alcanzar primero, es aquí donde aparecen las intenciones. Las intenciones pueden ser consideradas como un conjunto de planes para lograr los objetivos que constituyen los deseos.

Como se apunta en [GEORGE95], no es necesario que un sistema especificado en términos de creencias, deseos e intenciones sea diseñado por medio de estructuras de datos correspondientes a cada uno de estos componentes. Sin embargo tal diseño podría simplificar la construcción, mantenimiento y verificación del sistema resultante.

A modo de ejemplo se presenta una arquitectura abstracta con tres estructuras dinámicas representando cada una las creencias, los deseos y las intenciones del agente, junto con una cola de eventos. Se permiten operaciones de actualización y consulta sobre las tres estructuras. Los eventos que el sistema puede reconocer incluyen eventos internos y eventos externos.

La siguiente secuencia de acciones se ejecuta en un ciclo infinito. Al inicio de cada ciclo, se lee la cola de eventos y se obtiene una lista de opciones. Seguidamente se selecciona un subconjunto de opciones a ser adoptadas y se las agrega a la estructura de intenciones. Si existe una intención para ejecutar una acción atómica en ese punto, el agente la ejecuta. Cualquier evento externo que ha ocurrido durante este tiempo se agrega a la cola de eventos. Los eventos internos se agregan apenas ocurren. Posteriormente el agente modifica las estructuras de intención y deseos de acuerdo al resultado de la ejecución de la intención.

1.5.3.2 Arquitecturas para agentes Reactivos

Una arquitectura para agente reactivo es aquella que no incluye ningún tipo de modelo simbólico central del mundo, y no utiliza razonamiento simbólico complejo por ejemplo:

1.5.3.2.1 Subsumption Architecture

Consiste en una jerarquía de comportamientos de logro de tareas [BROOKS86]. Cada comportamiento 'compite' con otros para ejercer control sobre el agente. Capas menores representan comportamientos de tipo más primitivo, (tal como evitar obstáculos, por ejemplo), y tienen precedencia sobre las capas superiores de la jerarquía. El sistema resultante es, en términos de porcentaje de computación que necesitan realizar, extremadamente simple, sin un razonamiento explícito del tipo encontrado en los sistemas simbólicos de inteligencia artificial.

1.5.3.2.2 PENGI

Chapman y Agre observaron en [CHAPMA86], que la mayoría de las actividades cotidianas son 'rutinas', en el sentido de que se requiere poco o ningún nuevo razonamiento. La mayoría de las tareas, una vez aprendidas, pueden ser desarrolladas de una manera rutinaria, con poca variación. Agre propuso que una arquitectura de agente eficiente podría ser basada en la idea de 'running arguments'. En forma cruda, la idea es que, como la mayoría de las decisiones son rutinarias, pueden ser codificadas a través de una estructura de bajo nivel (tal como un circuito digital), que sólo necesita actualización periódica para manejar nuevos tipos de problemas. Su enfoque fue ilustrado con el sistema PENGI. PENGI es un juego de computadora simulado con un sistema central controlado utilizando el esquema arriba definido.

1.5.3.2.3 Situated Autómata

En el paradigma *situated automata* [KAELBL86] un agente se especifica en términos declarativos. Esta especificación se compila luego a una máquina digital que satisface la especificación declarativa. Esta máquina digital puede operar de una manera 'time-bounded'; no realiza ningún tipo de manipulación simbólica y de hecho ninguna expresión simbólica se representa en la máquina. La lógica utilizada para especificar un agente es esencialmente una lógica modal de conocimiento. Un agente se especifica en términos de dos componentes: percepción y acción. Dos programas se utilizan para sintetizar a los agentes: RULER se emplea para especificar el componente referente a la percepción de un agente; mientras que GAPPS para el componente referente a la acción del agente.

RULER toma como entrada tres componentes:

> "Una especificación de la semántica de las entradas del agente; un conjunto de hechos estáticos; y una especificación del estado de transiciones del mundo. El programador luego especifica la semántica deseada para la salida, y el compilador... sintetiza un circuito cuyas salidas tendrán la semántica correcta. ... Todo el conocimiento declarativo se reduce a un circuito bastante simple"

GAPPS toma como entrada un conjunto de reglas de reducción de objetivos, (esencialmente reglas que codifican información acerca de cómo los objetivos pueden ser satisfechos), y un objetivo de alto nivel, y genera un programa que puede ser traducido a un circuito digital para realizar el objetivo. Nuevamente, el circuito generado no representa o manipula expresiones simbólicas; toda la manipulación simbólica se realiza en tiempo de compilación.

1.5.3.2.4 Arquitectura de Red de agentes.

Pattie Maes ha desarrollado una arquitectura de agentes en la cual un agente se define como un conjunto de módulos de competencia [MAES91]. Estos módulos asemejan ligeramente el comportamiento de la arquitectura subsumption. Cada módulo es especificado por el diseñador en términos de pre- y post- condiciones, y un nivel de activación, que da una indicación real de la relevancia del módulo en una situación particular. Cuanto más alto es el nivel de activación de un módulo, hay mayor posibilidad de que ese módulo influenciará el

comportamiento del agente. Una vez especificado un conjunto de módulos de competencia se compilan en una red de activación esparcida, en la cual los módulos se enlazan unos a otros de manera definida por las pre- y post- condiciones. El resultado de la ejecución puede ser un comando a una unidad efectora o tal vez el aumento del nivel de activación de un módulo sucesor.

Existen obvias similitudes entre una arquitectura de red de agentes y arquitecturas de redes neuronales. Tal vez la diferencia clave es que es difícil especificar cual es el significado de un nodo en una red neuronal; sólo tiene un significado en el contexto de la red en sí. A pesar de que los módulos de competencia se definen en términos declarativos, sin embargo, es mucho más fácil especificar cual es su significado.

1.5.3.3 Arquitecturas para Agentes Híbridos

Muchos investigadores [GEORGE87, FERGUS92, BURMEI92] han sugerido que ni un enfoque completamente deliberativo ni uno completamente reactivo es adecuado para construir agentes. Argumentaron el caso de sistemas híbridos que intentan unir los enfoques deliberativos y reactivos.

Un enfoque obvio es construir un agente compuesto por dos subsistemas: uno deliberativo, que contiene un módulo simbólico del mundo, que desarrolla planes y efectúa decisiones de la manera propuesta por la inteligencia artificial simbólica; y uno reactivo, que es capaz de reaccionar a eventos que ocurren en el ambiente sin necesitar un razonamiento complejo. A menudo, al componente reactivo se le da cierto grado de precedencia sobre el deliberativo, para proveer una pronta respuesta a eventos ambientales importantes. En una arquitectura tal, los sistemas de control del agente se arreglan en una jerarquía, con las capas más altas tratando con información de mayor nivel de abstracción. Así, por ejemplo, las capas inferiores pueden mapear datos crudos (monitoreados) directamente a los efectores de salida, mientras que las capas superiores tratan con objetivos a largo plazo.

1.5.3.3.1 PRS

De la misma manera que IRMA, el PRS [GEORGE87] es una arquitectura basada en creencias, deseos e intenciones, que incluye una biblioteca de planes, así como una explícita representación simbólica de las creencias, deseos e intenciones. Las creencias son hechos, tanto acerca del mundo exterior como del estado interno del sistema. Estos hechos son expresados mediante la clásica lógica de primer orden. Los deseos son representados como comportamientos del sistema (antes que como una representación estática de estados de objetivos). Una librería de planes de un PRS contiene un conjunto de planes parcialmente elaborados, llamados áreas de conocimiento (*knowledge areas, KAs),*cada uno de los cuales se asocia con una condición de invocación. Estas condiciones determinan cuando el KA debe ser activado. Los KAs pueden ser activados de una manera dirigida por objetivos o dirigida por datos; los KAs pueden ser también reactivos, permitiendo que el PRS responda rápidamente a cambios en su ambiente. El conjunto de KAs actualmente activos en un sistema representan sus intenciones. Estas varias estructuras de datos son manipuladas por un sistema intérprete, que es responsable de la actualización de las creencias, invocando KAs y ejecutando acciones.

1.5.3.3.2 TouringMachines

La arquitectura consiste de subsistemas de percepción y acción que realizan la interfaz directamente con el ambiente del agente, y de tres capas de control contenidas en un framework de control que media entre las capas. Cada capa es un proceso independiente productor de actividad que se ejecuta continuamente [FERGUS92].

La capa reactiva genera cursos potenciales de acción en respuesta a eventos que ocurren demasiado rápido como para que otras capas los traten. Está implementada como un conjunto de reglas situación-acción, en el estilo de la arquitectura subsumption.

La capa de planeamiento construye planes y selecciona acciones para ejecutar a fin de satisfacer los objetivos del agente. Esta capa consta de dos componentes: un planeador y un mecanismo foco de atención. El planeador integra la generación del plan y la ejecución, y utiliza una biblioteca de planes parcialmente elaborados, junto con un mapa topológico del mundo, a fin de construir planes que lograrán el principal objetivo del agente. El propósito del

mecanismo foco de atención es limitar el porcentaje de información con los que el planeador debe tratar y de esta forma aumentar su eficiencia. Logra esto filtrando información relevante procedente del ambiente.

La capa de modelado contiene representaciones simbólicas del estado cognitivo de otras entidades dentro del ambiente del agente. Estos modelos son manipulados a fin de identificar y resolver conflictos entre objetivos – situaciones donde el agente ya no puede lograr sus objetivos, como resultado de una interferencia inesperada.

Las tres capas son capaces de comunicarse unas con otras, (vía paso de mensajes), y están contenidas en un framework de control. El propósito de este framework es mediar entre las capas y, en particular, tratar con acciones conflictivas propuestas por las diferentes capas. El framework de control logra esto utilizando reglas de control.

1.5.3.3.3 COSY

La arquitectura COSY [BURMEI92] es un BDI (creencias, deseos e intenciones) híbrido que incluye elementos tanto de PRS como de IRMA. La arquitectura tiene cinco componentes principales: (i) sensores; (ii) actuadores; (iii) comunicaciones; (iv) cognición; e (v) intención. Los primeros tres componentes son directos: los sensores reciben entradas perceptibles no comunicativas, los actuadores permiten al agente realizar acciones no comunicativas y el componente comunicaciones permite al agente enviar mensajes. De los dos componentes restantes, el componente intención contiene objetivos a largo plazo, actitudes, responsabilidades y los elementos de control que toman parte en el razonamiento y la toma de decisiones del componente cognición, y el componente cognición es responsable de mediar entre las intenciones del agente y sus conocimientos acerca del mundo, y de elegir un acción apropiada para realizar. Dentro del componente cognición se encuentra la base de conocimiento que contiene las creencias del agente y tres componentes procedimentales: un componente de ejecución de script, un componente de ejecución de protocolo y un componente de razonamiento, decisión y reacción. Un script es una fórmula o plan para lograr

un objetivo. Los protocolos son diálogos que representan frameworks de cooperación. El componente de razonamiento, decisión y reacción es tal vez el componente clave de COSY. Está hecho de un número de otros subsistemas y es estructurado tal como PRS y IRMA. Se mantiene una agenda que contiene un número de scripts activos. Estos scripts pueden ser invocados de una manera dirigida por objetivos o de una dirigida por datos. Un componente filtrador selecciona entre scripts en competición para la ejecución.

1.5.3.3.4 Composicional

En una arquitectura composicional todas las funcionalidades son diseñadas como una serie de componentes estructurados jerárquicamente, que interactúan basados en tareas [BRAZIE97a]. Las tareas se caracterizan en términos de sus entradas, sus salidas y su relación con otras tareas. La interacción y cooperación entre componentes, entre componentes y el mundo externo, y entre componentes y usuarios se especifica en términos de intercambio de información, secuenciamiento de información y dependencias de control. Los componentes en sí pueden ser de cualquier complejidad y pueden realizar cualquier función de dominio.

Modelos de tareas definen la estructura de arquitecturas composicionales: los componentes en una arquitectura composicional están directamente relacionados a tareas en una (des)composición de tareas. En una arquitectura composicional son especificados y modelados explícitamente los siguientes elementos:

(1) Una (Des)Composición de tareas: por cada tarea en una jerarquía de tareas un conjunto de subtareas puede ser especificada;
(2) Intercambio de Información: se especifica como links de información entre componentes. Cada link de información relaciona la salida de un componente con la entrada de otro;
(3) Secuenciamiento de Tareas: se modela explícitamente dentro de los componentes como conocimiento de control de tarea. El conocimiento de control de tareas incluye no sólo conocimiento de qué tarea debe ser activada, cuando y cómo, sino también conocimiento sobre información de control asociado con la activación de

la tarea y el porcentaje de esfuerzo que puede permitirse para lograr un objetivo dado.

(4) Delegación de Subtareas: durante la adquisición del conocimiento una tarea como un todo se modela. Durante el proceso de modelado se toman decisiones como por ejemplo: qué tarea es desarrollada mejor por cuál agente. Este proceso, que en general también puede ser llevado a cabo en tiempo de ejecución, resulta en la delegación de (sub)tareas a partes envueltas en la ejecución de la tarea; y

(5) Estructuras de Conocimiento: durante la adquisición del conocimiento una estructura apropiada para el dominio del conocimiento debe ser proyectada. El significado de los conceptos utilizados para describir un dominio y las relaciones entre los conceptos y grupos de conceptos, debe ser determinado. Los conceptos se requieren para identificar objetos distinguibles en un dominio, pero además para expresar los métodos y estrategias empleadas para realizar la tarea.

1.5.3.3.5 BDI Composicional

En esta arquitectura el modelo genérico de un agente con arquitectura composicional es refinado en un modelo BDI genérico racional, en el cual el agente es capaz de razonamiento explícito acerca de sus creencias, deseos e intenciones. El modelo BDI Composicional [BRAZIE97, b] está basado en un análisis de las tareas desarrolladas por un agente BDI. Tal análisis de tareas resulta en una composición (jerárquica) de tareas, que es la base para un modelo composicional.

En la arquitectura composicional el modelo genérico establece las siguientes tareas necesarias para el agente:

1. control de sus propios procesos,
2. cumplimiento de sus tareas propias,
3. manejo de su interacción con el mundo,
4. manejo de su comunicación con otros agentes,
5. mantenimiento de información sobre el mundo, y
6. mantenimiento de información sobre otros agentes.

En la arquitectura BDI Composicional, cada una de las tareas anteriores es refinada descomponiéndolas en tres componentes:

1. creencias del agente
2. deseos del agente
3. intenciones del agente

Entonces la jerarquía de tareas presentada en el modelo genérico se extiende agregando los tres componentes anteriores. Por ejemplo, la tarea control de sus propios procesos se subdivide en las siguientes subtareas: determinación de creencias, determinación de deseos y determinación de intenciones.

El resultado es un agente BDI más específico en el cual las dependencias entre creencias, deseos e intenciones se hacen explícitas.

1.5.3.4 Arquitectura Adoptada

Del análisis realizado en la sección anterior se puede notar que existe una gran diversidad de ideas en cuanto a la forma de modelar la arquitectura de un agente. A partir de la esencia de cada una se pretende analizar aquella o aquellas que contemplan las características presentes en la definición de agente que se ha adoptado.

En una arquitectura para agentes reactivos, el interés se centra en el modelado de la reactividad, la autonomía y la interacción con el ambiente, a través de sensores y efectores, dejando de lado las características de inteligencia a partir de una representación interna de los conocimientos. Por tanto con una arquitectura de este tipo podría ser imposible capturar todas las características que se consideran de importancia para el modelado de agentes.
Algo más interesante se presenta en la arquitectura para agentes deliberativos, en cuyas diferentes propuestas, además de poder modelar las características de interacción con el ambiente y la autonomía, aparecen el componente pro-actividad, y una característica que se

considera de mucha importancia: el componente de nociones mentales. Asociados con este último aparecen la racionalidad y el aprendizaje.

Con una arquitectura para agentes híbridos se puede integrar todas las características de ambas arquitecturas (reactiva y deliberativa). El resultado es una arquitectura más específica, que modela en forma explícita ciertas características como ser: descomposición de tareas, delegación de tareas, estructuras mentales, etc.
De entre las analizadas en la sección anterior, la arquitectura BDI y la BDI Composicional, abarcan de manera más exhaustiva las características presentes en la definición de Wooldridge y Jennings. Sin embargo, al ser más general la arquitectura BDI, puede ser más adecuada para el análisis posterior.

1.5.4 Agentes vs. Objetos: Una Comparación

En esta sección se presenta una comparación entre los paradigmas Orientado a Objetos y de Agentes, identificando para ello similitudes, así como principales diferencias. Esta comparación servirá como punto de partida para una evaluación de la aplicabilidad del paradigma de objetos al modelado de agentes.

Como se explicó en la sección anterior, un objeto es una entidad que encapsula un estado y una colección de métodos, correspondientes a operaciones que pueden ser ejecutadas sobre ese estado. Los métodos son típicamente invocados como resultado de un mensaje enviado al objeto [BOOCH94, JACOB97, RUMB91].

Por otra parte, un agente es una entidad autónoma, social, reactiva, pro-activa, que posee nociones mentales, racionalidad y aprendizaje [WOOLDR95a]. El comportamiento pro-activo puede ser representado en términos de una biblioteca de planes de la cual el agente puede seleccionar alguno. Los agentes razonan acerca de sus creencias a fin de seleccionar el plan que podría lograr satisfacer sus objetivos. Debido a que los agentes operan en un ambiente complejo, formado por otros agentes, entidades software no necesariamente agentes y usuarios humanos, requieren de mecanismos y comportamiento que les permitan comunicarse con estas entidades. Al coexistir con otros agentes, es necesario algún mecanismo o propiedad que pueda diferenciar a un agente de otro; por tanto cada agente debe poseer una identidad

que sea independiente de su contenido. Un agente además de lo anterior debe mantener ciertas propiedades que determinan su estado, a partir del cual puede determinar las acciones a realizar y el comportamiento a seguir.

Se pueden encontrar ciertas analogías entre agentes y objetos, por ejemplo:

- Estado: los agentes y los objetos tienen un estado interno basado en su historia e influenciado por su comportamiento.
- Identidad: tanto los objetos como lo agentes poseen identidad que no está determinada únicamente por los valores de sus atributos.
- Encapsulación: los agentes y los objetos pueden proteger u ocultar datos y comportamiento.
- Comportamiento: los objetos y los agentes pueden reaccionar recibiendo o enviando mensajes.

Como lo indica Elizabeth Kendall [KENDAL96], se puede considerar a los agentes como objetos autónomos de tiempo real que poseen un comportamiento pro-activo.

Pero existen varias diferencias. Entre ellas se pueden destacar:

- Cada objeto ejecuta secuencialmente métodos, mientras que un agente ejecuta un razonamiento autónomo que le permite elegir de forma “inteligente” una alternativa.
- Los objetos poseen datos y comportamiento, mientras que los agentes poseen además creencias, deseos e intenciones.
- Un agente reflexivo ejecuta un razonamiento autónomo para seleccionar un plan, mientras que un objeto no razona.
- Un objeto simplemente responde a mensajes y eventos, mientras que los agentes no son sólo reactivos, sino también activos en el sentido de que son capaces de actuar dirigidos por objetivos sin un estímulo externo.

- Los agentes son flexibles, es decir, pueden cambiar su comportamiento en tiempo de ejecución; son capaces de lidiar con situaciones no previstas o inesperadas, etc.

En síntesis, y como lo sostiene [SHOHAN93], los agentes pueden ser interpretados como objetos autónomos e inteligentes, que están equipados con capacidades de conocimiento y razonamiento para satisfacer varios objetivos, en un determinado entorno.

1.6 METODOLOGÍAS ORIENTADAS A OBJETOS

En esta sección se revisa varias de las metodologías orientadas a objetos propuestas para desarrollar sistemas multi-agentes

1.6.1 Metodología

No existe un consenso entre los diversos autores sobre el concepto de metodología.
Una metodología puede definirse, en un sentido amplio, como un conjunto de métodos y técnicas que ayudan en el desarrollo de un producto software, tal como lo señala Rumbaugh:

> "*Una metodología de ingeniería de software es un proceso para la producción organizada del software, empleando para ello una colección de técnicas predefinidas y convenciones en las notaciones. Una metodología se presenta normalmente como una serie de pasos, con técnicas y notaciones asociadas a cada paso... Los pasos de la producción del software se organizan normalmente en un ciclo de vida consistente en varias fases de desarrollo"*[RUMBAUGH91].

De un modo mas general una metodología podría definirse como "un conjunto de conceptos para poder abstraer el dominio del problema, una notación para representar esos conceptos, una serie de pasos y procedimientos a seguir y un conjunto de activos, bienes o aciertos generados".

1.6.2 Campo de la Metodología Orientada a Objetos

En el mundo de desarrollo de software, se tiende cada vez mas hacia la programación orientada a objetos, un nuevo paradigma de programación que aporta grandes ventajas al software generado, como son el acercamiento entre el concepto usuario y el resultado programado o la disponibilidad a la reutilización del código generado.

Todas las metodologías distinguen estas tres fases:

- Análisis
- Diseño
- Implementación

Según Piattini [Piattini96], se puede observar un cambio "filosófico" entre las metodologías clásicas de análisis y diseño estructurado y las de orientación al objeto. En las primeras se examinan los sistemas desde el punto de vista de las funciones o tareas que deben realizar, tareas que se van descomponiendo sucesivamente en otras tareas más pequeñas y que forma los bloques o módulos de las aplicaciones. En la orientación a objetos, por su parte, cobra mucha más importancia el aspecto de "modelado" del sistema, examinando el dominio del problema como un conjunto de objetos que interactúan entre si.

En las metodologías tradicionales se produce una dicotomía entre los dos elementos constituyentes del sistema: funciones que llevan a cabo los programas y datos que se almacenan en archivos o bases de datos. Sin embargo la orientación al objeto propugna un enfoque unificador de ambos aspectos que se encapsulan en los objetos.

Se pueden identificar dos enfoques en las metodologías orientadas a objetos[FICHMAN y KEMERER92]:

- "Revolucionarias" o "puras", que extienden la orientación orientada al objeto como un cambio profundo que convierte a las metodologías estructuradas en obsoletas.
- "Sintéticas " y "evolutivas", que piensan que el análisis y diseño estructurados constituyen la base para el desarrollo orientado a objetos, pudiéndose combinar elementos del análisis y diseño estructurados, con los de la orientación al objeto.

1.6.2.1 El Método de BOOCH

La Booch es una metodología de propósito general en la que se parte de que cada etapa no es un proceso aislado sinó que ha de interactuar con sus siguientes y precedentes en una especie de bucle del que se sale cuando se esté satisfecho con el modelo conseguido. En un principio se tienen una serie de objetos y clases que forman el sistema, a continuación se construye el modelo de interfaz y se examinan las relaciones entre las clases lo que, a su vez, genera la adición de nuevos interfaces que generarán nuevas relaciones iterándose hasta llegar al estado de refinamiento deseado. El método Booch proporciona un conjunto de herramientas gráficas y notaciones que ayudan a representar visualmente los modelos definidos en las fases de análisis y diseño. Algunas de ellas son:

-Diagramas de clase.
Se trata de una variación de los diagramas de entidad relación en los que se añaden nuevos tipos de relaciones como la herencia, instanciación y uso. Además permite agrupar las clases y relaciones en categorías para diagramas demasiado complejos.

-Diagramas de objeto.
En este tipo de gráfico de muestran los objetos y sus relaciones de forma dinámica mostrando la forma en la que los objetos se pasan mensajes entre ellos. Así mismo, en esto diagramas es posible representar la visibilidad de los objetos siendo ésta la que determina que objetos se pueden comunicar con otros.

-Diagramas temporales.
Muestran la secuencia temporal de creación y destrucción de objetos. Suelen ir acompañados de pseudocódigo en el que se explica el flujo de mensajes de control entre los objetos del sistema.
-Diagramas de transición de estados.
Permiten definir como las instancias de las clases pasan de un estado a otro a causa de ciertos eventos y que acciones se desencadenan de esos cambios de estado.

-Diagramas de módulo y proceso.
En Booch, en la fase de implementación, es posible representar mediante estos gráficos la parte física del sistema, es decir, se puede mostrar como se van a almacenar internamente las

clases y objetos, relaciones entre módulos en tiempo de compilación, procesos, dispositivos y las comunicaciones entre ellos.

Ventajas

- Es una metodología de propósito general.
- Herramientas y notaciones comprensibles.
- Proporciona una gran cantidad de información sobre las semánticas de los objetos.

Inconvenientes

- Notaciones poco precisas.
- Poca ayuda para el desarrollador novel.
- Herramientas orientadas al texto más que a los gráficos.

1.6.2.2 OMT (OBJECT MODELING TECHNIQUE).

La OMT, al contrario que la Booch, divide el proceso de desarrollo en tres partes aisladas: análisis, diseño e implementación. A su vez cada una de estas fases consta de otras subtareas como son los modelos de objetos, dinámico y funcional del análisis y el de sistema y objetos en el diseño.

- Modelo de objetos

En esta primera parte del análisis se forma una primera imagen del modelo de clases del sistema con sus atributos y las relaciones entre ellas, usando para ello un diagrama entidad relación modificado en el que además de las clases y sus relaciones se pueden representar también los métodos.

- Modelo dinámico.

El modelo dinámico usa un grafo para representar el comportamiento dinámico de cada clase, es decir, el comportamiento de estas ante cada evento que se produce en el sistema. Un evento desencadenará un cambio de estado en la clase que se traducirá en una modificación de los atributos o relaciones de ésta.

- Modelo funcional.

Muestra que es lo que el sistema ha de hacer mediante un diagrama de flujo de datos, sin entrar en la secuencia temporal en la que los procesos se ejecutan. El modelo funcional puede revelar nuevos objetos y métodos que se pueden incorporar en los dos modelos anteriores. Por eso se dice que el método OMT es iterativo.

- Diseño del sistema.

Se centra en la parte física del sistema como la descomposición de éste en subsistemas, el tipo de entorno en el que se va a ejecutar, el manejo de recursos y el almacenamiento de datos.

-Diseño de los objetos.

Determina que operaciones van a realizar los métodos y profundiza incluso que los algoritmos que va a usar. Se escogen los distintos tipo de representación de datos y se subdivide todo en módulos que pasarán a formar parte de la implementación.

- Implementación.

En esta fase se convierte finalmente el diseño de objetos en código.

Ventajas

- Proporciona una serie de pasos perfectamente definidos al desarrollador.
- Tratamiento especial de la herencia.
- Facilita el mantenimiento dada la gran cantidad de información que se genera en el análisis.

Inconvenientes

- Hay pocos métodos para encontrar inconsistencias en los modelos.
- Interacción de objetos no soportada explícitamente en ninguna herramienta gráfica.
- Al ser un análisis iterativo es difícil de saber cuando comenzar con el diseño.

1.6.2.3 FUSION (Método de Coleman).

El método Fusion toma los mejores aspectos de cada uno de las metodologías anteriores. Por ejemplo de la OMT toma los modelos de objetos, de la Booch sus gráficos de visibilidad, incluso toma los cuadros de pre y postcondición de las metodologías estructuradas. Además, aunque se pueda aplicar a cualquier tipo de desarrollo, esta especialmente diseñado para proporcionar técnicas y herramientas al analista de sistemas industriales y de producción. Las distintas tareas de las que consta, tanto en el análisis como en el diseño son:

- Modelo de objetos.

Es similar al usado en la OMT.

- Modelo operacional.

Es análogo al modelo funcional de la OMT, sin embargo se ha prescindido de los diagramas de flujo de datos debido a que en la práctica estos se han demostrado inapropiados y se han adoptado las especificaciones de precondición y postcondición, propias de los métodos formales, para especificar que es lo que el sistema hace.

- Modelo de ciclo de vida.

Se usan notaciones textuales de gramáticas para representar la operaciones que se realizan durante la vida de una entidad.

- Gráficos de interacción de objetos.

Muestra como los objetos se comunican entre ellos mediante el paso de mensajes.

- Gráficos de visibilidad.

Determina como un objeto conoce la existencia de otro, y si es exclusiva o compartida.

- Descriptores de clase

Se trata de un refinamiento de las definiciones anteriores de las clases.

Ahora de detallan sus atributos, junto con los tipos de datos de que se tratan, y los métodos para manejar esos atributos.

- Gráficos de herencia.

Mediante unos gráficos sencillo se muestra las especializaciones y generalizaciones de cada una de las clases que intervienen en el sistema.

Ventajas

- Proporciona una gran cantidad de herramientas para comprobar la validez y consistencia de los modelos realizados en el análisis y diseño.
- El proceso de modelado es sistemático están definidos todos sus pasos.
- Es posible modelar el flujo de control y la creación y destrucción dinámica de objetos.
- No sólo es apropiado para sistemas de gestión.

Inconvenientes

- A veces es un método excesivamente riguroso.
- No es un proceso iterativo

1.6.2.4 La Unificación de Métodos

Después de explicar varias metodologías orientadas de objetos, se puede observar que, a pesar de algunas notables diferencias existentes entre una y otras, todas persiguen el mismo objetivo: la obtención de un software de calidad dentro del paradigma de la orientación al objeto.

Por lo tanto se puede llegar fácilmente a la conclusión de que se intentará llegar a una unificación de los diferentes métodos.

He aquí una serie de motivos que también contribuyen a esta tendencia:

- El esfuerzo por la estandarización y la convergencia en la orientación al objeto.
- Los avances en herramientas CASE OO.
- El interés por el modelado de negocios mediante objetos.

Comenzaron a surgir: UML, OPEN, ...

1.6.2.5 UML (Unified Modelating Languaje)

Proviene sobre todo de: BOOCH93, OMT93 y OOSE.

La idea de unificación pretende en un principio estabilizar el caos existente en las metodologías orientadas al objeto, así como la aparición de un modelo más rico, producto del intercambio de experiencias en las tres metodologías génesis de UML.

"Es un lenguaje para especificar, construir, visualizar y documentar ingenio software, cuyo alcance pretende cubrir loa conceptos de BOOCH, OMT, OOSE, resultando un lenguaje simple, común y ampliamente utilizable por otras metodologías".

UML no es una metodología OO, sino una notación universal. El utilizar un lenguaje de modelado estándar le da un valor añadido.

Es un lenguaje para representar los modelos que se obtienen a partir de la aplicación de cualquier metodología orientada a objetos.

UML no fuerza a utilizar ninguna metodología concreta, porque presupone que distintos dominios de problemas conducen a diferentes métodos de análisis y diseño.
En su versión 1. 0 los principales elementos son: un metamodelo y una semántica, una notación gráfica y un conjunto de recomendaciones.

1.6.2.6 OPEN

OPEN fue creada en un principio a partir de una mezcla de algunas metodologías de segunda generación (MOSES, SOMA y "The Firesmith Method").
Es esencialmente un armazón para la tercera generación de métodos de desarrollo software, en la orientación al objeto, suministrando un gran soporte para el proceso de modelado mediante el uso de modelos de ciclo de vida, mediante una captura de requisitos y ofreciendo la habilidad de modelar o construir agentes inteligentes.

OPEN también toma conceptos de BON, Martín/Odell, OBA, RDD, ROOM, UML y otros.

Ofrece un conjunto de principios para el modelado de todos los aspectos del desarrollo software: ciclo de vida, tareas, técnicas y modelado del lenguaje.

OPEN extiende el concepto de metodología, no solo incluyendo un modelo de procesos, sino también suministrando líneas para construir versiones del método que se ajusten a las necesidades del dominio industrial, organizaciones individuales.

Los elementos principales de esta metodología son:

- Ciclo de vida o metamodelo.
- Técnicas.
- Representación.

Se ha conseguido una metodología que comprenda, al menos, un conjunto de técnicas, más un modelo de ciclo de vida y una representación.

Las actividades de OPEN tienen tareas que se ejecutan y se completan gracias a las tareas.

1.6.2.7 El Proceso Unificado Rational

El Proceso Unificado Rational [KRUCH98] es una metodología propuesta por la Rational Software Corporation, desarrollada por Jacobson, Booch y Rumbaugh, y adopta la notación UML [BRJ97]. Constituye una metodología iterativa e incremental, dirigida por Casos de Uso y centrada en la arquitectura. En un ciclo de vida iterativo e incremental, el desarrollo procede como una serie de iteraciones que evolucionan hasta el sistema final. Cada iteración consiste de los siguientes componentes de proceso: análisis de los requerimientos, análisis, diseño, implementación y prueba.

1.7 METODOLOGÍAS ORIENTADAS A AGENTES

En esta sección se revisa varias de las metodologías orientadas a agentes propuestas para desarrollar sistemas multi-agentes.

1.7.1 Campo de la Metodología Orientada a Agentes

Luego de una revisión del estado del arte de varias metodologías orientadas a agentes, se puede concluir que existen dos enfoques principales que siguen las mismas:

- Extensión de Metodologías Orientadas a Objetos, a fin de incluir aspectos relevantes de los Agentes.
- Extensión de Metodologías de Ingeniería del Conocimiento (KE, Knowledge Engineering), tratando de aprovechar las técnicas de la KE para modelar características cognitivas de los Agentes.

1.7.1.1 Extensión de Metodologías Orientadas a Objetos

A nuestro parecer, se pueden citar varias razones que justifican la extensión de las metodologías orientadas a objetos.
Entre estas razones tenemos:

- existen similitudes entre el paradigma orientado a objetos y el paradigma orientado a agentes. Como establece Shoham, [SHOHAM93], los agentes pueden ser considerados objetos activos, objetos con estados mentales.
- el uso común de los lenguajes orientados a objetos para la implementación de los agentes
- la popularidad de las metodologías orientadas a objetos

1.7.1.1.1 Análisis y Diseño Orientado a Agentes, de Burmeister

Burmeister [BURMEIS96] define tres modelos para analizar un sistema agente: el modelo de agente, que se limita al agente y a su estructura interna (creencias, planes, objetivos); el modelo organizacional, que describe el relacionamiento entre agentes (herencia y roles en la organización); y el modelo de cooperación que describe la interacción entre los agentes.

Los pasos del proceso de desarrollo de cada modelo son:

- Modelo de Agentes: se identifican los agentes y sus ambientes utilizando una extensión de las tarjetas CRC (Clases-Responsabilidades-Colaboraciones) a fin de incluir creencias, motivaciones, planes y atributos de cooperación.
- Modelo Organizacional: propone la identificación de los roles de cada agente y la elaboración de diagramas utilizando la notación OMT [RUMBAU91] para la jerarquía de herencia y las relaciones entre los agentes.
- Modelo de Cooperación: se identifican las cooperaciones y los modelos de cooperación y los tipos de mensajes que se intercambian entre agentes y además se analizan los protocolos utilizados.

1.7.1.1.2 Técnicas de Modelado de Agentes para Sistemas de agentes BDI

Este método [KINNY96] define dos niveles principales (externo e interno) para el modelado de agentes BDI.

El nivel externo consiste de la descomposición del sistema en agentes y la definición de sus interacciones. Esto se lleva a cabo a través de dos modelos:

- el modelo de agentes, para describir la jerarquía de relaciones entre agentes y las relaciones entre agentes concretos; y
- el modelo de interacción, para describir las responsabilidades, servicios e interacciones entre agentes y sistemas externos.

El nivel interno lleva a cabo el modelado de cada clase de agente BDI a través de tres modelos:

- el modelo de creencias, que describe las creencias a cerca del ambiente;
- el modelo de objetivos, que describe los objetivos y eventos que un agente puede adoptar o responder; y
- el modelo de planes, que describe los planes que una gente puede utilizar para lograr sus objetivos.

El proceso de desarrollo del nivel externo se inicia con la identificación de los roles (funcionales, organizacionales, etc.) del dominio de la aplicación, a fin de identificar los

agentes y organizarlos en una jerarquía de clases de agentes descrito utilizando una notación parecida a la OMT [RUMBAU91]. Luego las responsabilidades asociadas a cada rol son identificadas, junto con los servicios proveídos y utilizados para cumplir con las responsabilidades. El siguiente paso es la identificación de las interacciones necesarias para cada servicio. Finalmente se colecta la información en un modelo de instancia de agente.

El desarrollo del nivel interno, se inicia con el análisis de las diferentes formas (planes) de lograr un objetivo. Los planes para responder a un evento o lograr un objetivo se representan utilizando una notación similar a los statecharts de Harel [HAREL87], pero agregando la noción de falla de un plan. Finalmente, las creencias del agente acerca de objetos del ambiente se modelan y representan utilizando la notación OMT [RUMBAU91].

1.7.1.1.3 Método para Multi-Agentes basado en Escenarios (MASB)

Este método [MOULIN97] está destinado a ser aplicado a sistemas multi-agentes (MAS) en el campo del trabajo cooperativo. La fase de análisis consta de las siguientes actividades:

- Descripción de Escenarios: identificación de los principales roles tanto de los humanos (usuarios del sistema) y los agentes software como de los objetos del ambiente y los escenarios típicos, en lenguaje natural.
- Descripción funcional de los roles: descripción de los roles de los agentes utilizando diagramas de comportamiento que describen los procesos, la información relevante y la interacción entre los agentes.
- Modelado conceptual del dato y el mundo: modelado del dato y el conocimiento utilizado por el agente empleando diagramas entidad-relación (o diagramas orientadas a objetos) y diagramas de ciclo de vida de entidades.
- Modelado de la interacción sistema-usuario: simulación y definición de diferentes interfaces para la interacción humano-máquina en todos los escenarios.

La fase de diseño consiste de la siguientes actividades:

- Descripción de la arquitectura y el escenario del sistema MAS: selección de los escenarios a ser implementados y los roles de los agentes en dichos escenarios.

- Modelado de Objetos: refinamiento del modelado del mundo definido en el análisis, definición de jerarquías, atributos y procedimientos.
- Modelado de Agentes: especificación como estructuras de creencias, de los elementos definidos en el modelado conceptual del análisis. Una notación gráfica se propone para describir el proceso de decisión de un agente, tomando en cuenta creencias, planes, objetivos e interacciones.
- Finalmente se establecen dos pasos aún no desarrollados: modelo conversacional y validación general del diseño del sistema.

1.7.1.1.4 Metodología Orientada a agentes para el Modelado de Empresas

Esta metodología [KENDAL96] propone la combinación de metodologías orientadas a objetos (OOSE [JACOBS97]) y metodologías de modelado de empresas IDEF (Integration Definition for Function modelling) [FIPS93]. Los modelos identificados son:

- Modelo Funcional: describe las funciones (inputs, outputs, mecanismos y control) utilizando diagramas $IDEF_0$ que incluyen la selección de los métodos posibles dependiendo del input y el control.
- Modelo Use Case: describe a los actores envueltos en cada función, utilizando la notación de use cases de OOSE [JACOBS97]
- Modelo Dinámico: este modelo está destinado a analizar las interacciones entre objetos. Los use cases se representan en diagramas de transición de eventos
- El sistema Orientado a Agentes: esta compuesto de:
    - Identificación de agentes: los actores de los use cases son identificados como agentes.
    - Protocolos de coordinación o scripts: se describen en diagramas de estados.
    - Invocación de planes: diagramas de secuencia extienden a los diagramas de transición de estados a fin de incluir condiciones para indicar cuando se invoca a un plan.
    - Creencias, Sensores y Efectores: los inputs de las funciones deben ser modelados como creencias u obtenidos de los objetos vía sensores, y los objetivos satisfechos deben ser modelados como cambios a las creencias vía efectores.

1.7.1.1.5 Ingeniería de Software Orientada a Agentes (AOSE)

El proceso de AOSE [GROSSE95] sigue los métodos de diseño Orientado a Objetos y consiste de tres partes, que son:

- Análisis del Sistema Orientado a Agentes: en esta etapa se identifican a los agentes que exhiben el comportamiento deseado del sistema. Cada agente tiene su conjunto propio único de capacidades que lo caracterizan.
- Diseño Orientado a Agentes: se define la integración de los agentes identificados en la etapa anterior y los roles de los mismos. Un aspecto importante de los agentes en un sistema multi-agente es que usualmente éstos están organizados en jerarquías. Este modelo de organización también se define en la etapa de diseño.
- Programación Orientada a Objetos: Se elige una arquitectura apropiada de agentes, y los agentes se implementan de acuerdo a la especificación de las dos etapas anteriores.

1.7.1.1.6 Ingeniería de Sistemas Multi-Agentes (MASE)

Esta metodología [DELOAC95] sigue los siguientes pasos: Diseño a nivel del Dominio, Diseño a nivel del Agente, Diseño de Componente y Diseño del Sistema.

Los pasos en el diseño a nivel del Dominio son:

1. Identificar los tipos de agentes
2. Identificar las posibles interacciones entre los tipos de agentes
3. Definir protocolos de coordinación para cada tipo de coordinación

Los pasos específicos de diseño a nivel de agentes son:

1. Mapear las acciones identificadas en la conversación de los agentes a componentes internos
2. Definir estructuras de datos identificadas en la conversación de agentes. Esas estructuras de datos representan inputs o outputs de los agentes

3. Definir estructuras de datos adicionales, internas al agente. Estas estructuras de datos representan los flujos de datos entre los componentes en la arquitectura.

Diseño de componentes: Una vez que la arquitectura del agente ha sido definida, los componentes especificados deben ser diseñados.

Los pasos específicos del diseño del sistema incluyen:

1. Seleccionar los tipos de agentes que son necesarios
2. Determinar el número de agentes requeridos para cada tipo y definir:
    a. La ubicación física o dirección de los agentes
    b. Los tipos de conversaciones que los agentes serán capaces de mantener
    c. Cualquier otro parámetro definido en el dominio.

1.7.1.1.7 Una metodología a nivel de organización para el diseño de multi-agentes

Esta metodología [GUTKNE95] es un proceso de MAS (Sistema Multi-Agente) en cuatro pasos: descripción de la estructura organizacional, definición de las características de los agentes, diseño individual de los agentes y la implementación del sistema final.

1. Definición de grupos y extracción de las inter-relaciones: el resultado es un conjunto de estructuras a ser especificadas y sus interdependencias.
2. Especificación de las estructuras de grupo: este paso define las estructuras de grupo como un conjunto de roles y funciones de admisión, además de una grafo de interacción.
3. Especificación de las características de los agentes: este paso identifica un conjunto de características de los agentes. Esta fase es la conjunción entre la especificación organizacional y el diseño de agentes. Cada característica define a un modelo de agente específico o arquitectura y una vista organizacional expresada en el modelo.
4. Diseño individual de agentes e Implementación: se realiza siguiendo cualquier metodología orientada a agentes (Ejemplo: AOSE, MASE, etc.)

1.7.1.2 Extensión de Metodologías de Ingeniería del Conocimiento

Según [GLASER96] las metodologías de ingeniería del conocimiento pueden proveer una buena base para el modelado de sistemas multi-agentes, dado que ellos se ocupan del desarrollo de sistemas basados en el conocimiento. Y como los agentes poseen características cognitivas, estas metodologías pueden proveer las técnicas para el modelado de estos agentes.

La definición del conocimiento de un agente puede ser considerada como un proceso de adquisición de conocimiento, sólo este proceso se estudia en estas metodologías.

1.7.1.2.1 La metodología CoMoMAS

Glaser [GLASER96] propone una extensión de la metodología CommonKADS [SCHREI94] para el modelado de sistemas multi-agentes. Se definen los siguientes modelos:

- Modelo de Agentes: este es el modelo principal de la metodología y define la arquitectura y el conocimiento del agente, que es clasificado como un conocimiento social, cooperativo, de control, cognitivo y reactivo.
- Modelo de Capacidad: describe la competencia cognitiva y reactiva del agente. Distingue entre tareas, resolución de problemas y conocimiento reactivo.
- Modelo de Tarea: describe la descomposición de las tareas y detalla si la tarea se resuelve por el usuario o el agente
- Modelo de Cooperación: describe la cooperación entre los agentes
- Modelo del Sistema: define los aspectos organizacionales de la sociedad de agentes junto con los aspectos arquitecturales de los agentes.
- Modelo de Diseño: reúne los modelos previos a fin de operacionalizarlos junto con los requerimientos no funcionales.

1.7.1.2.2 La metodología MAS-CommonKADS

Esta metodología [IGLESI97] extiende el modelo definido en CommonKADS, agregando técnicas de metodologías orientadas a objetos (OOSE [JACOBS97], OMT [RUMBAU91]).

La metodología inicia con una fase de conceptualización que es una fase informal destinada a colectar los requerimientos del usuario y obtener una primera descripción del sistema desde el punto de vista del usuario. Para este propósito se utilizan técnicas de OOSE [JACOBS97]

La metodología incluye los siguientes modelos:

- Modelo de Agentes: describe las características principales de los agentes, incluyendo las capacidades de razonamiento, habilidades (sensores/efectores), servicios, objetivos, etc.
- Modelo de Tarea: describe las tareas llevadas a cabo por el agente y la descomposición de tareas.
- Modelo de Capacidades: describe el conocimiento que necesita el agente para llevar a cabo las tareas.
- Modelo de Coordinación: describe las conversaciones entre los agentes, esto es sus interacciones, protocolos y capacidades requeridas.
- Modelo de Organización: describe la organización en la cual el sistema multi-agente será introducido y la organización de la sociedad de agentes.
- Modelo de Comunicación: detalla las interacciones humano-agente y los factores humanos para desarrollar estas interfaces.
- Modelo de Diseño: reúne los modelos previos y se subdivide en tres submodelos:
    - Diseño de la aplicación: composición o descomposición de los agentes del análisis
    - Diseño de la arquitectura: diseño de los aspectos relevantes de la red de agentes
- Diseño de la plataforma: selección de la plataforma de desarrollo de los agentes para cada arquitectura de agente.

CAPÍTULO 3
MARCO METODOLÓGICO

1.8 DISEÑO DE LA INVESTIGACIÓN

El presente estudio se enmarca en el campo de los conceptos de tipo descriptivo, experimental, de corte longitudinal. Descriptivo en razón de que se señalan las Metodologías Orientadas a Objetos y a Agentes. Experimental, por cuanto se realiza la construcción de un sistema aplicando las metodologías seleccionadas tanto Orientada a Objetos como a Agentes. Corte Longitudinal, por cuanto se aplica las diferentes fases de desarrollo de un sistema agente.

1.9 POBLACIÓN Y MUESTRA

La población esta constituida por miembros de la Escuela Politécnica del Ejército Sede Latacunga, que poseen experiencia en la aplicación de Metodologías para el desarrollo de Sistemas. En el presente caso fueron seleccionados profesores y personal administrativos que tienen una formación superior en el campo Informático, que prestan sus servicios en la facultad de Sistemas, Organización de Sistemas y Ciencias Básicas. Por ser la población pequeña se selecciona toda.

TIPO	POBLACIÓN	CRITERIO DE SELECCIÓN
Profesores	12	Conocimientos y experiencias en la Aplicación de Metodologías para la construcción de Productos Software
Administrativos	8	

TABLA 3.1: Muestra de Trabajo

1.10 HIPÓTESIS

Las Metodologías Orientadas a Objetos con relación a las Metodologías Orientadas a Agentes facilitarán la construcción de Agentes.

1.11 OPERACIONALIZACIÓN DE VARIABLES

Las variables a utilizarse se muestran en la Tabla # 2 Operacionalización de Variables.

MATRIZ TECNOLOGICA DE LA INVESTIGACION					
HIPÓTESIS	**VARIABLES**	**DEFINICION CONCEPTUAL**	**INDICADORES**	**INDICES**	**INSTRUMENTOS**
			Definiciones		
	Variable		Métodos, Técnicas y	Aplicación de Metodologías	
	Dependiente		Procedimientos	Orientadas a Objetos	
	Construcción de	Creación de un Sistema	Ciclo de Vida	Aplicación de Metodologías	Entrevista/Encuesta
	un Agente	Inteligente con autonomía	Software	Orientadas a Agentes	
	Inteligente		Hardware		
Las Metodologías Orientadas			Herramientas de Desarrollo		
a Objetos con relación a las	Variable				
Metodologías Orientadas a	Independiente				
Agentes facilitarán la	Metodologías	Es un proceso para la	Definiciones	Grado de conocimiento	
construcción de Agentes.	Orientadas	producción organizad de	Métodos, Técnicas y	sobre Metodologías Orien-	
	a Objetos	un Software, mediante la	Procedimientos	tadas a Objetos	
		utilización del paradigma	Ciclo de Vida		Entrevista/Encuesta
		Objeto.	Software	Grado de conocimiento	
	Metodologías	Es un proceso para la	Hardware	sobre Metodologías Orien-	
	Orientadas	producción organizad de	Herramientas de Desarrollo	tadas a Agentes	
	a Agentes	un Software, mediante la			
		utilización de un nuevo			
		paradigma Agentes			

TABLA 3.2: Operacionalización de Variables

1.12 PROCEDIMIENTOS GENERALES

La ejecución de la investigación tiene el siguiente mecanismo:
Investigación de campo

MÉTODO: Inductivo
TÉCNICAS: Encuesta
INSTRUMENTO: Cuestionario/Entrevista

1.13 INSTRUMENTOS DE LA INVESTIGACIÓN

Según la naturaleza de la investigación la encuesta/entrevista es el instrumento mas apropiado para la recolección de datos; es aplicada a dos tipos de miembros de la muestra: profesores y administrativos.

Ver Anexo 1: Formato de la Encuesta.

1.14 VALIDACIÓN DE LOS INSTRUMENTOS

Para convalidar tanto la validez como la confiabilidad de la encuesta/entrevista, como instrumento de la investigación, se realizo una aplicación piloto y se solicito accesoria para su verificación y confiabilidad de resultados.

Los instrumentos contienen preguntas abiertas y cerradas, con las que se puede asegurar el criterio que se plantea y la respuesta al mismo.

1.15 PROCEDIMIENTO PARA LA RECOLECCIÓN DE DATOS

- Elaboración de Instrumentos para levantar la información
- Pilotaje y corrección de instrumentos
- Aplicación de las encuestas

1.15.1 RESUMEN DE LAS ENCUESTAS PARA LOS INDICADORES DE LA VARIABLE INDEPENDIENTE

Variable Independiente: Metodologías Orientadas a Objetos

Indicador: Definiciones

Pregunta 1:
Dé un concepto sobre objeto

		Frecuencia	**Porcentaje**
Válidas	Correctas	20	100.0
	Incorrectas	0	0.0

Pregunta 2:
Cuáles son las propiedades de un objeto?

		Frecuencia	**Porcentaje**
Válidas	Correctas	20	100.0
	Incorrectas	0	0.0

Tabla de contingencia Pregunta 1 y 2

		Cuales son las propiedades de un objeto?		Total
		Correctas	Incorrectas	
De un concepto sobre objeto?	Correctas	20	0	20
	Incorrectas	0	0	0
Total		20	0	20

Resumen de Pregunta 1 y 2

Pregunta	**Porcentaje**
P1	100%
P2	100%
Promedio	100%

TABLA 3.3: Análisis de Resultados, variable independiente: indicador 1

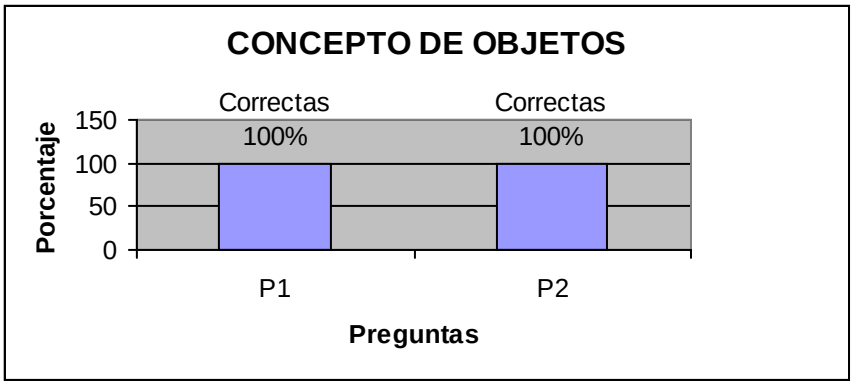

Figura 3.1: Gráfico de Barras de la variable independiente: indicador 1

Interpretación:

Se aprecia un excelente conocimiento sobre las definiciones correspondientes al paradigma Objeto y sus propiedades.

Variable Independiente: Metodologías Orientadas a Agentes

Indicador: Definiciones

Pregunta 3:
Dé un Concepto de Agente

		Frecuencia	Porcentaje
Válidas	Correctos	14	70.0
	Incorrectos	6	30.0
Total		20	100.0

Pregunta 4:
Conoce los tipos de Agentes?

P4.a: Conoce sobre el tipo de agente de software deliberativo?

		Frecuencia	Porcentaje
Válidas	Correctos	14	70.0
	Incorrectos	6	30.0
	Total	20	100.0

P4.b: Conoce sobre el tipo de agente de software reactivo?

		Frecuencia	Porcentaje
Válidos	Correctos	15	75.0
	Incorrectos	5	25.0
	Total	20	100.0

P4.c: Conoce sobre el tipo de agente de software embebido?

		Frecuencia	Porcentaje
Válidas	Correctos	16	80.0
	Incorrectos	4	20.0
	Total	20	100.0

P4.d: Conoce sobre el tipo de agente de software híbrido?

		Frecuencia	Porcentaje
Válidas	Correctos	11	55.0
	Incorrectos	9	45.0
	Total	20	100.0

P4.e: Conoce sobre otros tipos de agentes de software?

		Frecuencia	Porcentaje
Válidas	Ninguno	20	100.0

Resumen Pregunta 4

		Frecuencia	Porcentaje
Válidas	Correctos	6	30.0
	Incorrectos	14	70.0
Total		20	100.0

Pregunta 5.
Seleccione las diferencias entre un objeto y un agente

P5.a: Los Objetos son flexibles, los agentes no

		Frecuencia	Porcentaje
Válidas	Correctos	15	75.0
	Incorrectos	5	25.0
Total		20	100.0

P5.b: El agente es reflexivo

		Frecuencia	Porcentaje
Válidas	Correctos	11	55.0
	Incorrectos	9	45.0
Total		20	100.0

P5.c: Los agentes son autónomos y los objetos no

		Frecuencia	Porcentaje
Válidas	Correctos	20	100.0
	Incorrectos	0	0.0
Total		20	100.0

Resumen Pregunta 5.

		Frecuencia	Porcentaje
Válidas	Correctos	10	50.0
	Incorrectos	10	50.0
Total		20	100.0

Tabla de contingencia: Pregunta 3 y 4

		Conoce los tipos de Agentes		Total
		Correcto	Incorrecto	
De un Concepto de Agente	Correcto	5	9	14
	Incorrecto	1	5	6
Total		6	14	20

Resumen de Pregunta 3, 4 y 5

Pregunta	Porcentaje
P3	70%
P4	30%
P5	50%
Promedio	50%

TABLA 3.4: Análisis de Resultados, variable independiente: indicador 2

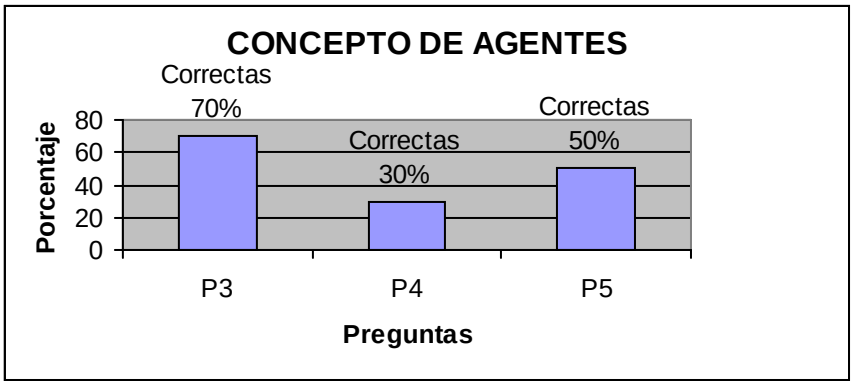

Figura 3.2: Gráfico de Barras de la variable independiente: indicador 2

Interpretación:

Un 70% de los encuestados conoce lo que es un agente; sin embargo, un bajo porcentaje puede distinguir los tipos de agentes de software que existen y sus diferencias con los objetos.

Variable Independiente: Metodologías Orientadas a Objetos

Indicador: Metodologías

Pregunta 6:
Conoce sobre la Metodologías Orientadas a Objetos?

		Frecuencia	Porcentaje
Válidas	SI	20	100.0
	NO	0	0.0
	DESCONOZCO	0	0.0
Total		20	100.0

Pregunta 7:
El Proceso Unificado Rational y BOOCH utilizan el Lenguaje de Modelado Unificado?

		Frecuencia	Porcentaje
Válidas	SI	20	100.0
	NO	0	0.0
	DESCONOZCO	0	0.0
Total		20	100.0

Resumen Pregunta 6 y 7

Pregunta	Porcentaje
P6	100%
P7	100%
Promedio	100%

TABLA 3.5: Análisis de Resultados, variable independiente: indicador 3

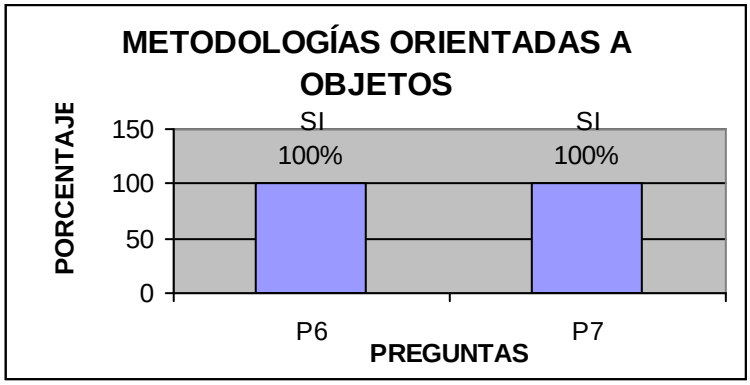

Figura 3.3: Gráfico de Barras de la variable independiente: indicador 3

Interpretación:

Todos los encuestados responden acertadamente que conocen las Metodologías Orientadas a Objetos e identifican tipos básicos de metodologías.

Variable Independiente: Metodologías Orientadas a Agentes

Indicador: Metodologías

Pregunta 8:
Conoce sobre la Metodologías Orientadas a Agentes?

		Frecuencia	Porcentaje
Válidas	SI	8	40.0
	NO	12	60.0
	DESCONOZCO	0	0.0
Total		20	100.0

Pregunta 9:
El Técnica de Modelado de Agentes para sistemas de Agentes BDI, utiliza los modelos de Creencias, Objetos y Planes?

		Frecuencia	Porcentaje
Válidas	SI	6	30.0
	NO	14	70.0
	DESCONOZCO	0	0.0
Total		20	100.0

Pregunta 10:
La Metodología MAS-CommonKADS se basa en los modelos de tareas, capacidades, comunicación?

		Frecuencia	Porcentaje
Válidas	SI	6	30.0
	NO	14	70.0
	DESCONOZCO	0	0.0
Total		20	100.0

Resumen Pregunta 8, 9y 10

Pregunta	Porcentaje
P8	40%
P9	30%
P10	30%
Promedio	33.3%

TABLA 3.6: Análisis de Resultados, variable independiente: indicador 4

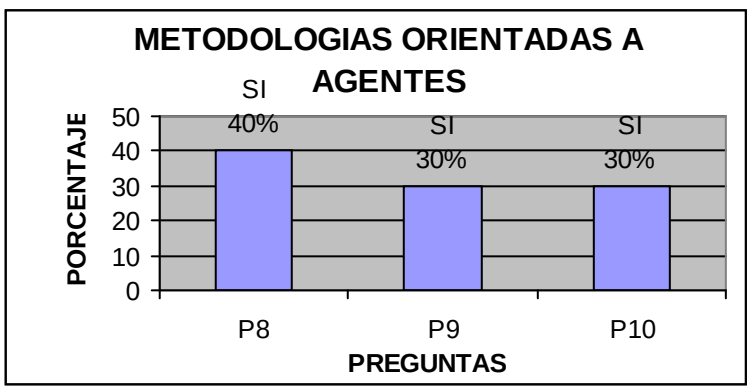

Figura 3.4: Gráfico de Barras de la variable independiente: indicador 4

Interpretación:

Muy pocas encuestados conocen sobre las Metodologías Orientadas a Agentes, así como los tipos de Metodologías.

Variable Independiente: Metodologías Orientadas a Objetos

Indicador: Herramientas

Pregunta 11:
Ha utilizado usted alguna herramienta de Software para el Modelado de Objetos?

		Frecuencia	Porcentaje
Válidas	SI	20	100.0
	NO	0	0.0
	DESCONOZCO	0	0.0
Total		20	100.0

Resumen Pregunta 11

Pregunta	Porcentaje
P11	100%
Promedio	100%

TABLA 3.7: Análisis de Resultados, variable independiente: indicador 5

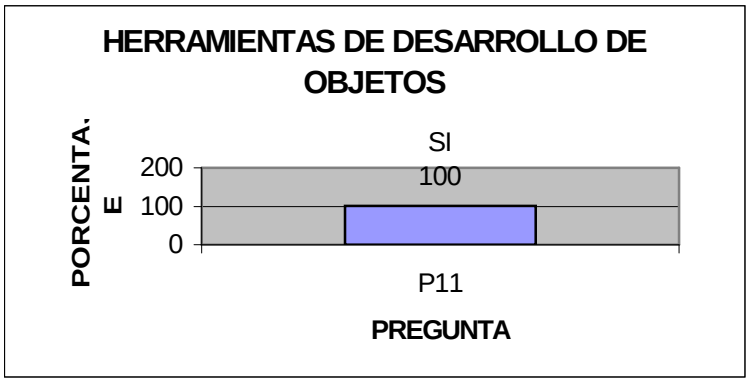

Figura 3.5: Gráfico de Barras de la variable independiente: indicador 5

Interpretación:

Todos los encuestados conocen herramientas para el modelado de Objetos.

Variable Independiente: Metodologías Orientadas a Agentes

Indicador: Herramientas

Pregunta 12:
Ha utilizado usted alguna herramienta de Software para el Modelado de Objetos?

		Frecuencia	Porcentaje
Válidas	SI	2	10.0
	NO	18	90.0
	DESCONOZCO	0	0.0
Total		20	100.0

Resumen Pregunta 12

Pregunta	Porcentaje
P12	10%
Promedio	10%

TABLA 3.8: Análisis de Resultados, variable independiente: indicador 6

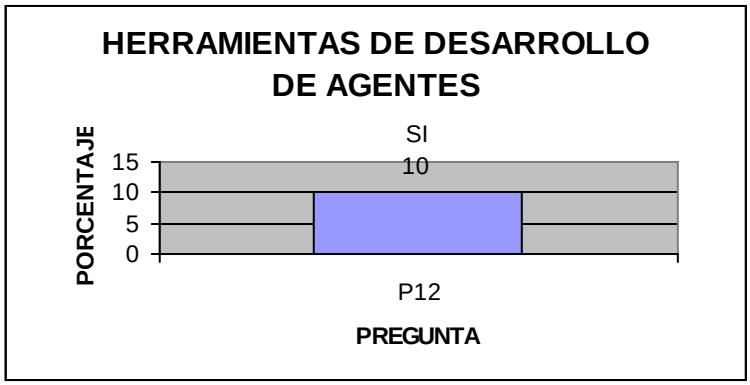

Figura 3.6: Gráfico de Barras de la variable independiente: indicador 6

Interpretación:

Poquísimos encuestados conocen herramientas para el modelado de Agentes.

1.15.2 RESUMEN DE LAS ENCUESTAS PARA LOS INDICADORES DE LA VARIABLE DEPENDIENTE

Variable Dependiente: Construcción de un Agente Inteligente

Indicador: Aplicación de Metodologías Orientadas a Objetos en Productos de Software

Pregunta 13:
Usted ha aplicado la Metodología Orientada a Objetos en la construcción de un Producto Software?

		Frecuencia	Porcentaje
Válidas	SI	20	100.0
	NO	0	0.0
Total		20	100.0

Resumen Pregunta 13

Pregunta	Porcentaje
P13	100%
Promedio	100%

TABLA 3.9: Análisis de Resultados, variable independiente: indicador 1

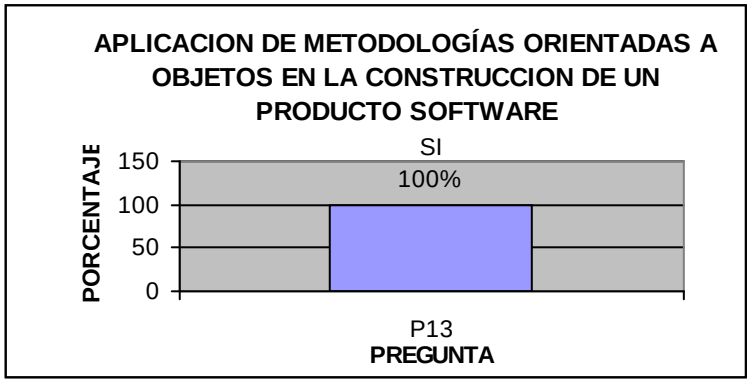

Figura 3.7: Gráfico de Barras de la variable dependiente: indicador 1

Interpretación:

Todos los encuestados alguna vez han aplicado la Metodología Orientada a Objetos en el desarrollo de un producto Software.

Variable Dependiente: Construcción de un Agente Inteligente

Indicador: Aplicación de Metodologías Orientadas a Objetos en Agentes de Software

Pregunta 14:
Usted ha aplicado la Metodología Orientada a Objetos en la construcción de un Agente Inteligente?

		Frecuencia	Porcentaje
Válidas	SI	2	10.0
	NO	18	90.0
Total		20	100.0

Resumen Pregunta 14

Pregunta	Porcentaje
P14	10%
Promedio	10%

TABLA 3.10: Análisis de Resultados, variable independiente: indicador 2

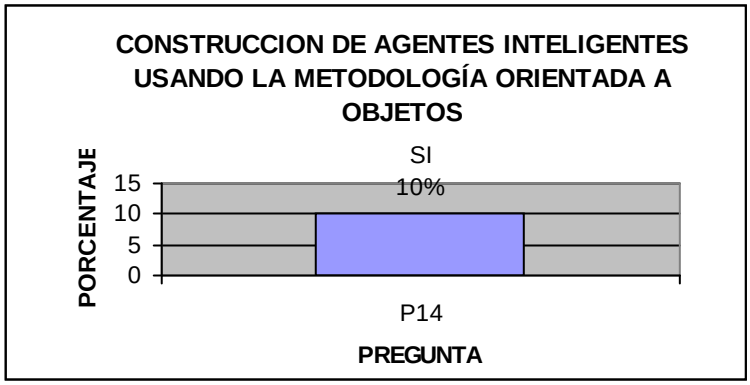

Figura 3.8: Gráfico de Barras de la variable dependiente: indicador 2

Interpretación:

Poquísimos encuestados han aplicado esta metodología en la construcción de un Agente Inteligente.

Variable Dependiente: Construcción de un Agente Inteligente

Indicador: Aplicación de Metodologías Agentes

Pregunta 15:
Usted ha aplicado la Metodología Orientada a Agentes en la construcción de un Agente Inteligente?

		Frecuencia	Porcentaje
Válidas	SI	2	10.0
	NO	18	90.0
Total		20	100.0

Resumen Pregunta 15

Pregunta	Porcentaje
P15	10%
Promedio	10%

TABLA 3.11: Análisis de Resultados, variable independiente: indicador 3

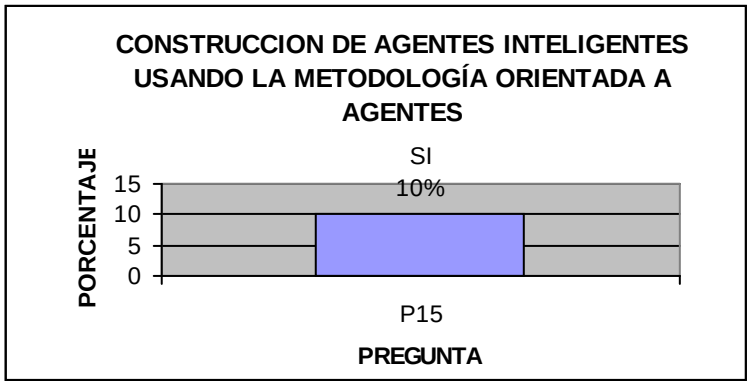

Figura 3.9: Gráfico de Barras de la variable dependiente: indicador 3

Interpretación:

Poquísimos encuestados han aplicado alguna Metodología Orientada a Agentes en la construcción de un Agente Inteligente.

1.16 RESULTADOS

1.16.1 VARIABLE INDEPENDIENTE

A continuación se indica los porcentajes de la variable independiente.

INDICADOR	PORCENTAJE
Definiciones de Objetos	100%
Metodologías de Objetos	100%
Herramientas de Objetos	100%
Definiciones de Agentes	50%
Metodologías de Agentes	33.3%
Herramientas de Agentes	10%

TABLA 3.12: Datos en porcentajes de la variable independiente

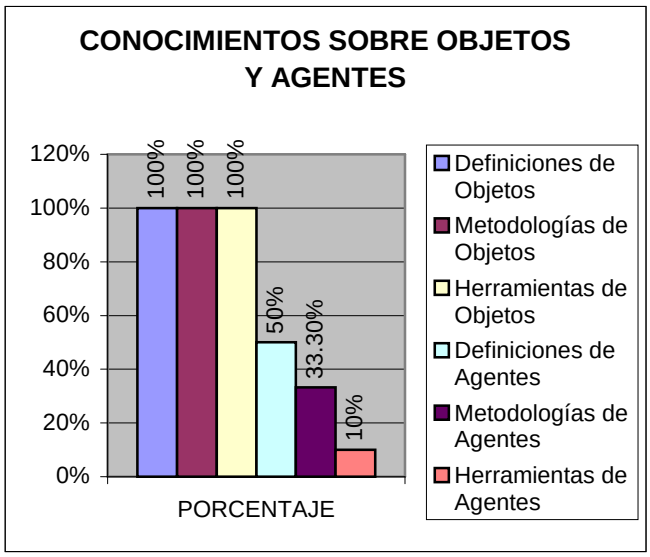

Figura 3.10: Gráfico de Barras de la variable independiente

Interpretación:

Las Definiciones sobre Objetos, Metodologías Orientadas a Objetos y Herramientas que permiten modelar Objetos son mucho mas conocidas y aplicadas que las de Agentes.

Además se desprende que se desconoce la interrelación existente entre Objetos y Agentes.

1.16.2 VARIABLE DEPENDIENTE

A continuación se indica los porcentajes de la variable dependiente.

INDICADOR	PORCENTAJE
Aplicación de Metodologías Orientadas a Objetos en Productos de Software	100%
Aplicación de Metodologías Orientadas a Objetos en Agentes de Software	10%
Aplicación de Metodologías Orientadas a Agentes	10%

TABLA 3.13: Datos en porcentajes de la variable dependiente

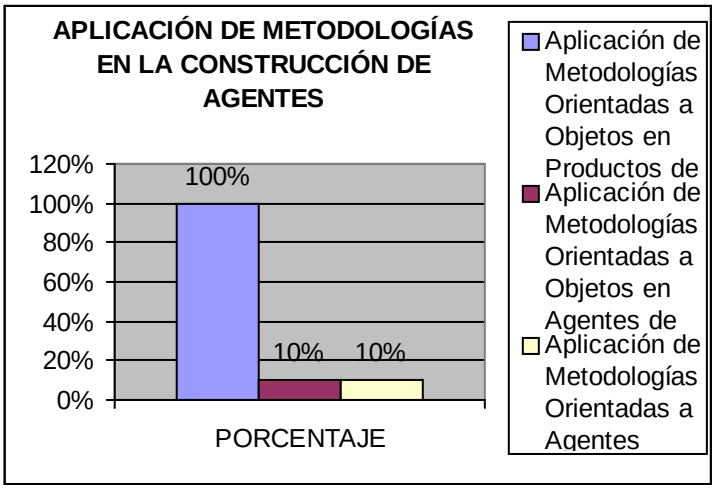

Figura 3.11: Gráfico de Barras de la variable dependiente

Interpretación:

Se demuestra que se aplica Metodologías Orientadas a Objetos en el desarrollo de Productos de Software pero no en el caso de aplicaciones de Metodologías Orientadas a Objetos en la construcción de Agentes de Software, así como tampoco con las Metodologías Orientadas a Agentes.

1.16.3 ANÁLISIS SOBRE LAS METODOLOGÍAS ORIENTADA A OBJETOS Y ORIENTADA A AGENTES.

Se parte de un trabajo experimental y con ello se verifica la hipótesis, tomando en cuenta que dentro de las estadísticas obtenidas en las encuestas muchos de los resultados casi nada aportan para la verificación de la misma, la razón es que dentro del grupo investigado, los profesionales de la ESPE, carecen o no tienen conocimientos sobre esta Metodología (Agentes).

Para comprobar la hipótesis se parte del análisis de las metodologías, elaborando un cuadro comparativo de las Metodologías, orientadas a objetos y a Agentes, identificando en cada una de las fases de desarrollo en la construcción de una Agente, las similitudes, ventajas y desventajas en sus distintas fases; este cuadro se elabora de acuerdo al trabajo realizado en esta tesis.

CUADRO COMPARATIVO ENTRE METODOLOGÍAS ORIENTADA A OBJETOS Y ORIENTADA A AGENTES

ORD.	FASE	METODOLOGÍA ORIENTADA A OBJETOS (Proceso Unificado Rational)	METODOLOGÍA ORIENTADA A AGENTES (MAS-commonKADS, BDI)	SIGNO
1	**Conceptualización** Empezar a identificar los requerimientos del Sistema 1. Diagramas de Casos de Uso 2. Diagramas de Secuencia	**1** **2**	**1** **2**	**0**
2	**Análisis** Determinación de requisitos del sistema 1. Entidades Estáticas del dominio de la aplicación 2. Entidades autónomas y dinámicas de la aplicación 3. Relaciones entre las entidades constituyentes 4. Interacción entre las entidades constituyentes	**1^{+}** **2^{-}** **3^{+}** **4^{+}**	**2** **3** **4^{-}**	**+**
3	**Diseño** Descripción de la arquitectura del sistema 1. Identificación y especificación interna de todos los módulos del sistema. 2. Identificación y especificación de relaciones entre los módulos	**1^{+}** **2^{+}**	**1** **2**	**+**
4	**Implementación y Pruebas** Módulos implementados y verificables 1. Implementar en un lenguaje 2. Verificación	**1** **2**	**1** **2**	**0**
5	**Operación y Mantenimiento** Entrega del producto y corrige posibles errores 1. Validación 2. Mantenimiento	**1** **2**		+

Este cuadro comparativo se lo realiza con respecto a la Metodología Orientada a Objetos

1^{+} Indica que abarca de manera completa el aspecto tratado en la fase de desarrollo de la metodología.
1^{-} Indica que abarca parcialmente el aspecto tratado en la fase de desarrollo de la metodología.

Ahora se verifica la hipótesis:

Hipótesis nula:

La Metodología Orientada a Objetos facilita la utilización de la Metodología Orientada a Agentes en la construcción de Agentes.

Hipótesis Alternativa:

La Metodología Orientada a Objetos no facilita la utilización de la Metodología Orientada a Agentes en la construcción de Agentes.

1.16.4 VERIFICACIÓN DE LA HIPÓTESIS

Con la utilización de la prueba del signo, con un nivel de significancia del 5%, se establece bajo la distribución binomial la probabilidad de aceptación o rechazo de la hipótesis nula. Se utiliza esta forma de verificación de la hipótesis por cuanto la muestra es pequeña, sabiendo que el número de items son cinco y el número de signos es tres, entonces:

n es el numero de signos, sean estos positivos o negativos, no se incluye el cero
n = 3
número de signos positivos = 3
número de signos negativos = 0

Fórmula de la Función de densidad probabilística binomial

$$P(x) = nCx * p^{x} * q^{n-x}$$

$$P(x) = \frac{n}{x!\,(n-x)!} * (0.5)^{x} * (0.5)^{n-x}$$

Donde:

P(x) es la probabilidad de que se presenten x signos positivos

x corresponde al evento de que se presente uno o varios signos positivos

nCx son las combinaciones que existen entre el número de signos, en este caso los el signo positivo.

p = 0.50

p probabilidad de que aparezcan signos + (Aceptación)

q probabilidad de que no aparezcan signos + (Fracaso)

Aplicando la Fórmula se obtiene:

x	**P(x)**
0	0.125
1	0.375
2	0.375
3	0.125

Distribución simétrica

La probabilidad de aceptar la hipótesis nula es de 0,875 es decir la probabilidad de tener tres o mas signos positivos es 0,125 que al comparar con el nivel de significancia 0,025 éste recae en la zona de aceptación, por tanto se puede afirmar que la Metodología Orientada a Objetos es mejor que la Metodología Orientada a Agentes.

1.16.5 RESULTADOS FINALES

Se desprende que la Metodología Orientada a Objetos (Proceso Unificado Rational) es conocida, aplicada y adoptada ampliamente, por las facilidades y utilidades que ofrece. La Metodología Orientada a Agentes no es conocida lo suficiente y tampoco ha sido explotada.

De los resultados obtenidos del análisis de las Metodologías Orientada a Objetos y Metodologías Orientadas a Agentes también se puede concluir que la Metodología Orientada a Objetos, en sus diferentes fases, facilita la utilización de la Metodología Orientada a Agentes en la construcción de Agentes.

CAPITULO 4
CONSTRUCCIÓN DE UN AGENTE INTELIGENTE

Para la construcción de un agente inteligente es necesario seguir los pasos de una metodología de software en todo el ciclo de vida de un producto software, el desarrollo de un agente, así como los resultados obtenidos.

En el caso de las metodologías de software se seleccionan tanto las orientadas a objetos como las orientadas a agentes, descritas en el capitulo 2.

Para el desarrollo de un agente de software se describe el prototipo y se aplica las metodologías seleccionadas en la construcción del mismo llegando a las pruebas, sus resultados y la discusión de los mismos.

4.1 PROPUESTAS METODOLÓGICAS SELECCIONADAS

En esta sección se presenta una descripción de las metodologías a ser utilizadas en el trabajo de investigación. En la sección 4.1.2, se presenta la metodología Orientada a Objetos, y en la sección 4.1.3, se presentan las metodologías Orientadas a Agentes.

4.1.1 INTRODUCCIÓN

Parece claro que, para que la tecnología de agentes pruebe ser una herramienta útil para la construcción de sistemas, es necesario adoptar o desarrollar técnicas de ingeniería de software adecuadas. De la revisión del estado del arte sobre las metodologías orientadas a agentes (ver capítulo 3), se puede concluir, que un importante esfuerzo investigador ha sido desarrollado por la comunidad científica internacional en estos últimos años.

Actualmente, la mayoría de las aplicaciones de agentes existentes son desarrolladas de una manera ad hoc [WOOLD95a]– con especificaciones a priori limitadas y siguiendo poco o ninguna metodología de diseño rigurosa. Esto, posiblemente, se deba a que las pocas metodologías orientadas a agentes existentes sean poco conocidas, o tal vez, difíciles de aplicar por un ingeniero de software con una formación estándar.

Un camino alternativo sería utilizar tecnologías no agentes para el modelado de estos sistemas. Ya que, como lo especificaban Wooldridge y Jennings [WOOLD95, a], no existe evidencia de que cualquier sistema desarrollado utilizando metodología orientada a agente, no pueda ser también desarrollado con la misma facilidad utilizando una metodología no orientada a agentes, creo que constituiría un aporte interesante para la disciplina, lograr algún tipo de experiencias y resultados, aplicando la metodología orientada a objetos (que ha probado tener éxito en el modelado de sistemas complejos) al diseño de sistemas basados en agentes.

Algunos investigadores ([KENDALL96, KINNY96]), señalan que existen diferencias entre los agentes y los objetos (por ejemplo: la flexibilidad del agente, comportamiento autónomo, evolución del comportamiento dinámico, etc.), que impiden que las metodologías orientadas a objetos sean directamente aplicables a los agentes. Sin embargo, estos autores se centran en metodologías Orientadas a Objetos estándar y no consideran que metodologías como, OOSE [JACOB97] y el Proceso Unificado Rational [KRUCH98], por ser (según sus autores) metodologías abiertas, ofrecen la posibilidad de ampliación. En mi percepción una posible ampliación podría, con ayuda de lenguajes orientados a objetos formales que permiten la especificación de objetos con comportamiento dinámico (ejemplo: dyOSL [SAAKE95]), cubrir la especificación de objetos autónomos, flexibles y con comportamiento dinámico evolutivo.

Vale la pena destacar que, si bien es verdadero que las técnicas de desarrollo para los sistemas de agentes están en su infancia, es también cierto que cualquier técnica rigurosa de desarrollo de software puede ser útil. Esto induce la necesidad de comparar diferentes experiencias de diseño de agentes, aplicando diferentes metodologías para identificar ventajas relativas. Es por ello, que en mi trabajo de verificación de la aplicabilidad de una metodología orientada a objetos en el diseño e implementación de agentes, además de construir un prototipo de agente basándome en una metodología orientada a objeto, también lo haré siguiendo metodologías orientadas a agentes.

4.1.2 CRITERIOS DE SELECCIÓN DE LAS METODOLOGÍAS

Para la selección de las distintas metodologías tanto orientadas a objetos como a agentes se deben tomar en cuenta si las metodologías cumplen con los siguientes parámetros:

Metodología de propósito general: Abarca los aspectos fundamentales para el desarrollo de cualquier producto software.
Ciclo de Vida: Toda metodología debe cubrir un ciclo de desarrollo de un producto software.
Interrelación en las etapas de desarrollo: Todas las etapas de un ciclo de vida deben estar enlazadas entre si.

Etapa de Conceptualización: Identifica los requerimientos de la aplicación de acuerdo con las necesidades del usuario y con que nivel de abstracción lo realiza.

Etapa de Análisis: Determina los requisitos del sistema de acuerdo con la etapa de conceptualización y con que nivel de abstracción lo realiza.

Etapa de Diseño: Describe la arquitectura del sistema y con que nivel de abstracción lo realiza.

Etapa de Implementación y Verificación: Colección de módulos y su verificación

Etapa de Operación y Mantenimiento: Operatividad del sistema y corrección de errores

Herramientas para el desarrollo: Si la metodología posee implementos para su desarrollo.

Notaciones comprensibles: Si la metodología maneja notaciones que son fáciles de entender y mantener así como métodos unificados, por ejemplo UML.

Metodología abierta: Si permite agregar otras características que la metodología no dispone.

Ecléctico: Si la metodología se base en otras tomando sus mejores características.

En el caso particular de las metodologías orientadas a agentes se deben tomar en cuenta parámetros específicos que son propios de dichas metodologías como son:

Características de Inteligencia del Agente: Si la metodología dispone y puede implementar nociones mentales.

Roles, Responsabilidades, Interacciones entre Agentes: Si la metodología cumple ciertas exigencias requeridas por los agentes.

4.1.2.1 SELECCIÓN DE LAS METODOLOGÍAS ORIENTADAS A OBJETOS

Para la selección de las metodologías orientadas a objetos se toman en cuenta las ventajas e inconvenientes que tiene cada una de las metodologías descritas en el capitulo 2.

En cuanto a los aspectos y características de una metodología que puede considerarse completa, viable, operativa y eficiente, se consideran los siguientes parámetros.

METODOLOGIAS	BOOCH	OMT	FUSION	PROCESO UNIFICADO RATIONAL
1. Metodología de propósito general para el desarrollo de Sistemas Agentes.	CUMPLE	CUMPLE	SISTEMAS DE GESTION	CUMPLE
2. Ciclo de Vida (Iterativo-Incremental)	NO CUMPLE	CUMPLE	NO CUMPLE	CUMPLE
3. Interrelación en las etapas de desarrollo de un producto	CUMPLE	NO CUMPLE	CUMPLE	CUMPLE

software				

Cuadro Comparativo de Metodologías Orientadas a Objetos(Continua)

METODOLOGIAS	BOOCH	OMT	FUSION	PROCESO UNIFICADO RATIONAL
4. Etapa de Conceptualización	CUMPLE	NO CUMPLE	CUMPLE	CUMPLE+
5. Etapa de Análisis	CUMPLE	CUMPLE	CUMPLE	CUMPLE+
6. Etapa de Diseño	CUMPLE	CUMPLE	CUMPLE	CUMPLE
7. Etapa de Implementación y Verificación	CUMPLE	CUMPLE	CUMPLE	CUMPLE
8. Etapa de Operación y Mantenimiento.	CUMPLE	NO CUMPLE	NO CUMPLE	CUMPLE
9. Herramientas para el desarrollo	CUMPLE-	NO CUMPLE	CUMPLE-	CUMPLE
10. Notaciones comprensibles	CUMPLE-	CUMPLE	CUMPLE	CUMPLE
11. Metodología abierta	NO CUMPLE	NO CUMPLE	NO CUMPLE	CUMPLE
12. Ecléctico	NO	NO	BOOCH, OMT, CRC	BOOCH, OMT, FUSION, OOSE, Otras.

Cuadro Comparativo de Metodologías Orientadas a Objetos(Continuación)

- Indica que abarca de manera parcial el aspecto tratado por ejemplo: Herramientas orientadas al texto mas que a los gráficas, Notaciones poco precisas.

+ Indica que abarca de manera completa el aspecto tratado por ejemplo: Nivel de abstracción elevado

Como metodología orientada a objetos se adopta la denominada Proceso Unificado Rational [KRUCH98] por cumplir con los parámetros señalados como indispensables en el cuadro anterior, también por ser, según sus autores y mi experiencia previa en el modelado de sistemas orientados a objetos, una metodología fruto de las mejores características y conceptos de varias metodologías orientadas a objetos (OMT, Booch, OOSE, etc.); y una metodología abierta, a la que será posible, si fuera necesario, agregar la especificación formal de los objetos en un lenguaje como el dyOSL propuesta en [SAAKE95].

4.1.2.2 SELECCIÓN DE LAS METODOLOGÍAS ORIENTADAS A AGENTES

Para la selección de las metodologías orientadas a agentes se toman en cuenta las ventajas e inconvenientes que tiene cada una de las metodologías descritas en el capitulo 2.

En cuanto a los aspectos y características de una metodología que puede considerarse completa, viable, operativa y eficiente, se consideran los siguientes parámetros:

EXTENSIÓN DE METODOLOGÍAS DE INGENIERÍA DEL CONOCIMIENTO

METODOLOGÍAS	COMOMAS	MAS-COMMONKADS
1. Metodología de propósito general para el desarrollo de Sistemas Agentes.	**CUMPLE**	CUMPLE
2. Ciclo de Vida	ESPIRAL	ESPIRAL
3. Interrelación en las etapas de desarrollo de un producto software	CUMPLE	CUMPLE
4. Etapa de Conceptualización	NO CUMPLE	CUMPLE
5. Etapa de Análisis	CUMPLE	CUMPLE+
6. Etapa de Diseño	CUMPLE	CUMPLE+
7. Etapa de Implementación y Verificación	NO ESPECIFICA	NO ESPECIFICA
8. Etapa de Operación y Mantenimiento.	NO CUMPLE	NO CUMPLE
9. Herramientas para el desarrollo	CUMPLE-	CUMPLE-
10. Notaciones comprensibles	CUMPLE	CUMPLE
11. Metodología abierta	NO CUMPLE	CUMPLE
12. Ecléctico	COMMONKADS	COMMONKADS OMT, OOSE.
13. Características de Inteligencia del Agente	CUMPLE	CUMPLE+

Cuadro Comparativo: Extensión de Metodologías de Ingeniería del Conocimiento

- Indica que abarca, de manera parcial, el aspecto tratado; por ejemplo: No tienen muchas herramientas para el desarrollo de agentes.

+ Indica que abarca, de manera completa, el aspecto tratado; por ejemplo: El nivel de abstracción es elevado y abarca, de manera mas profunda, la especificación de las características de inteligencia del agente.

Se adopta la metodología MAS-CommonKADS [IGLESI97], pertenece al grupo de metodologías que son extensión de metodologías de Ingeniería del Conocimiento. Primero, porque se considera muy interesante el hecho de que las primeras etapas de la metodología poseen un nivel de abstracción elevado, que permite la identificación de los componentes iniciales del sistema de forma general. Segundo, porque abarca de manera más profunda, la especificación de las características de inteligencia del agente, al utilizar para ello métodos y técnicas de Ingeniería del Conocimiento.

EXTENSIÓN DE METODOLOGÍAS ORIENTADAS A OBJETOS

METODOLOGIAS	BURMEISTER	BDI	MASB
1. Metodología de propósito general para el desarrollo de Sistemas Agentes.	**CUMPLE**	**CUMPLE**	**CUMPLE**
2. Ciclo de Vida	WATERFALL	WATERFALL	WATERFALL
3. Interrelación en las etapas de desarrollo de un producto software	CUMPLE	CUMPLE	CUMPLE
4. Etapa de Conceptualización	NO CUMPLE	NO CUMPLE	NO CUMPLE
5. Etapa de Análisis	CUMPLE	CUMPLE+	CUMPLE
6. Etapa de Diseño	CUMPLE	CUMPLE+	CUMPLE
7. Etapa de Implementación y Verificación	NO CUBRE	NO CUBRE	NO CUBRE
8. Etapa de Operación y Mantenimiento.	NO CUBRE	NO CUBRE	NO CUBRE
9. Herramientas para el desarrollo	CUMPLE-	CUMPLE-	CUMPLE-
10. Notaciones comprensibles	CUMPLE	CUMPLE	CUMPLE
11. Metodología abierta	NO CUMPLE	NO CUMPLE	NO CUMPLE
12. Ecléctico	OMT	OMT	NO
13. Roles, responsabilidades, interacciones entre agentes	CUMPLE	CUMPLE+	CUMPLE

14. Características de Inteligencia del agente: Creencías, Objetivos y Planes	CUMPLE	CUMPLE+	CUMPLE

Cuadro comparativo: extensión de Metodologías Orientadas a Objetos

METODOLOGIAS	MODELADO DE EMPRESAS	AOSE	MASE
1. Metodología de propósito general para el desarrollo de Sistemas Agentes.	**CUMPLE**	**CUMPLE**	CUMPLE
2. Ciclo de Vida	WATERFALL	WATERFALL	WATERFALL
3. Interrelación en las etapas de desarrollo de un producto software	CUMPLE	CUMPLE	CUMPLE
4. Etapa de Conceptualización	NO CUMPLE	NO CUMPLE	NO CUMPLE
5. Etapa de Análisis	CUMPLE	CUMPLE	CUMPLE
6. Etapa de Diseño	CUMPLE	CUMPLE	CUMPLE
7. Etapa de Implementación y Verificación	NO CUBRE	NO CUBRE	NO CUBRE
8. Etapa de Operación y Mantenimiento.	NO CUBRE	NO CUBRE	NO CUBRE
9. Herramientas para el desarrollo	CUMPLE	CUMPLE	CUMPLE
10. Notaciones comprensibles	CUMPLE	CUMPLE	CUMPLE
11. Metodología abierta	NO CUMPLE	NO CUMPLE	NO CUMPLE
12. Ecléctico	OOSE, IDEF	NO	NO
13. Roles, responsabilidades, interacciones entre agentes	CUMPLE-	CUMPLE	CUMPLE
14. Características de inteligencia del agente Creencias, Objetivos y Planes	CUMPLE-	CUMPLE-	CUMPLE-

Cuadro comparativo: extensión de Metodologías Orientadas a Objetos

- Indica que abarca, de manera parcial, el aspecto tratado: por ejemplo: las características de inteligencia del agente Creencias, Objetivos y Planes.

+ Indica que abarca, de manera completa, el aspecto tratado por ejemplo: La especificación de las características de inteligencia del agente: creencias, objetivos y planes; los roles, responsabilidades e interacciones entre agentes con más detalles

Se adopta la Metodología de Técnicas de Modelado de Agentes para Sistemas de agentes BDI [KINNY96], la que se considera como la más completa del grupo correspondiente a extensión de la Orientación a Objetos, debido a que abarca tanto los detalles referentes a estados mentales (deseos, creencias e intenciones), así como los roles y responsabilidades de los distintos agentes dentro del dominio de aplicación y las interacciones necesarias entre los agentes componentes del sistema.

4.1.3 METODOLOGÍAS ORIENTADAS A OBJETOS

El interés en las metodologías Orientadas a Objetos ha crecido rápidamente en los últimos años. Esto se puede deber al hecho de que el paradigma Orientado a Objetos ha demostrado poseer varias cualidades interesantes. Entre las más prominentes cualidades de un sistema diseñado con una metodología Orientada a Objetos se pueden encontrar las siguientes [JACOBS97]:

- La **Comprensión** del sistema es más fácil ya que la brecha semántica entre el sistema y la realidad es pequeña.
- Las **Modificaciones** al modelo tienden a ser locales ya que generalmente resultan de ítems individuales, que están representados por un objeto único.

En la sección 4.1.2.1 se presenta el Proceso Unificado Rational [KRUCH98] y en la sección 4.1.2.2 el lenguaje dyOSL [SAAKE95].

4.1.3.1 El Proceso Unificado Rational

El Proceso Unificado Rational [KRUCH98] es una metodología propuesta por la Rational Software Corporation, desarrollada por Jacobson, Booch y Rumbaugh, y adopta la notación UML (Unified Modeling Language) [BRJ97]. Constituye una
metodología iterativa e incremental, dirigida por Casos de Uso y centrada en la arquitectura. En un ciclo de vida iterativo e incremental, el desarrollo procede como una serie de iteraciones que evolucionan hasta el sistema final. Cada iteración consiste de los siguientes componentes de proceso: análisis de los requerimientos, análisis, diseño, implementación y test.

El Proceso Unificado Rational se estructura en dos dimensiones diferentes:

- El tiempo: división del ciclo de vida en fases e iteraciones.
- Componentes de Proceso: producción de un conjunto específico de componentes con actividades bien definidas.

Estructurar el proyecto a lo largo de la dimensión del tiempo envuelve la adopción de las siguientes fases:

- Inicio: especificación de la visión del proyecto.
- Elaboración: planear las actividades necesarias y los recursos requeridos; especificando las características y diseñando la arquitectura.
- Construcción: construir el producto como una serie de iteraciones incrementales.
- Transición: proveer el producto a la comunidad de usuarios (construir, entregar y entrenar).

Estructurar el proceso a lo largo de la dimensión de componentes incluye las siguientes actividades:

- Análisis de Requerimientos: descripción de lo que debería hacer el sistema.
- Diseño: arquitectura del sistema. Cómo se realizará el sistema en fase de implementación.
- Implementación: la producción del código que resultará en un sistema ejecutable.
- Test: la verificación del sistema completo.

La interacción entre ambas dimensiones se observa en la figura 4.1. Como se puede notar en la figura, cada fase incluye cierto flujo de trabajo correspondiente a cada uno de los componentes de proceso.

4.3.4.2.1 *Arquitectura del sistema.*

Una vista de la representación de la arquitectura del Sistema se tiene en la figura 4.2. Aquí se puede observar el hecho de que todo el desarrollo de las distintas vistas del sistema gira en torno a los Casos

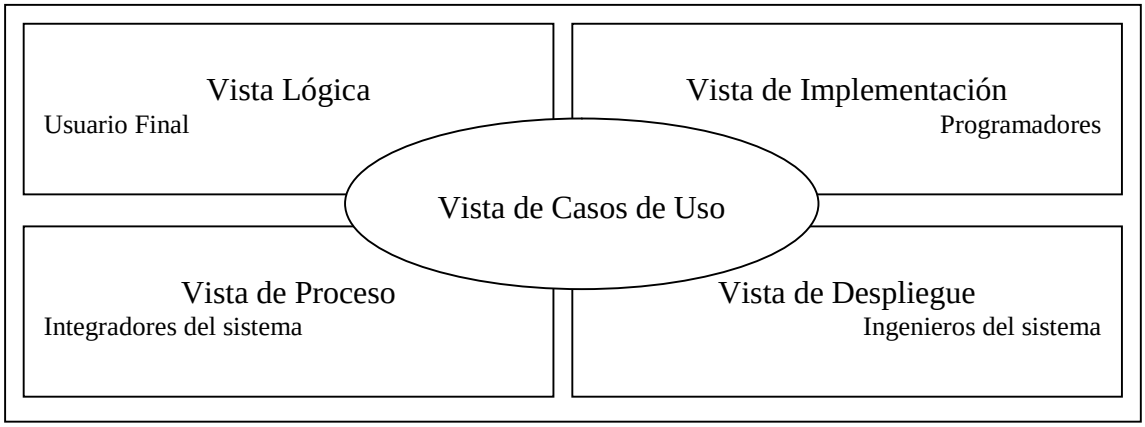

Figura 4.2: Representación de la Arquitectura del Sistema

de Uso.

Como se puede notar en la figura 4.2, la arquitectura de un sistema se organiza en términos de distintas capas. Cada capa representa una abstracción coherente y contiene una interfaz controlada y bien definida. Las vistas o capas identificadas son:

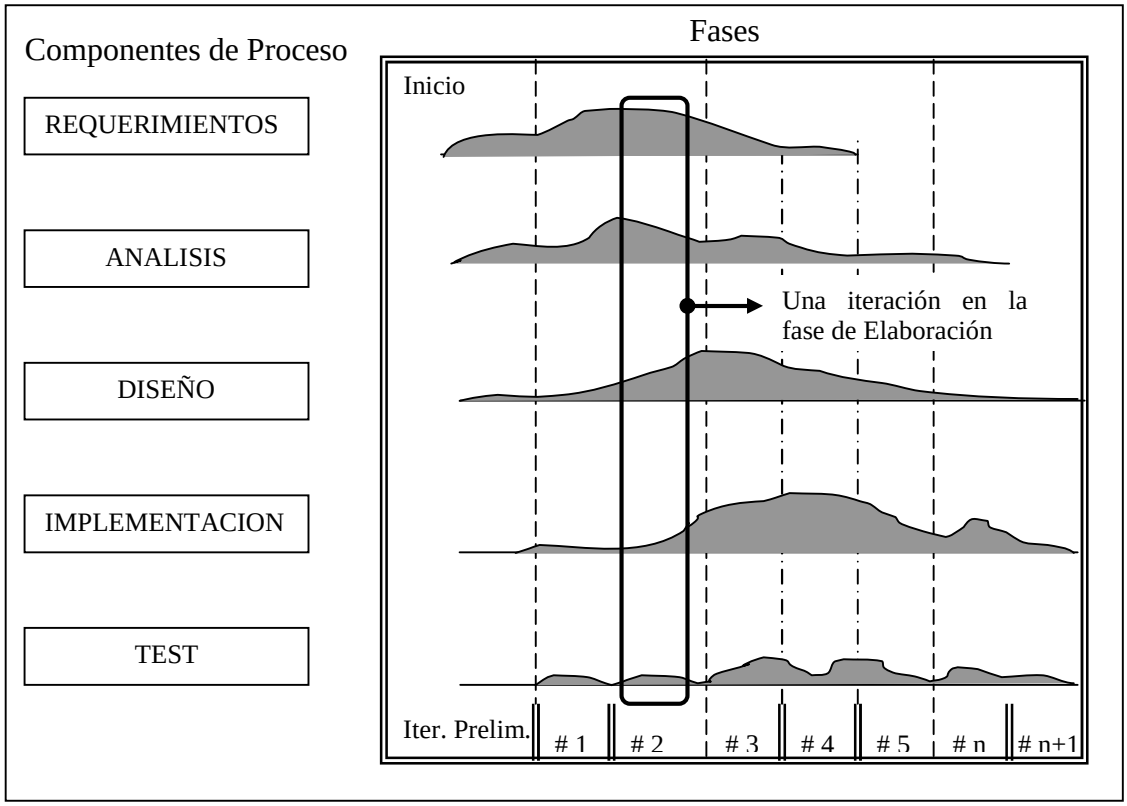

Figura 4.1: Interacción entre las dimensiones Tiempo y Componentes de Proceso

- La vista de *Casos de Uso* describe al sistema como un conjunto de transacciones desde el punto de vista de actores externos.
- La vista *Lógica* contiene una colección de paquetes, clases y relaciones.
- La vista de *Procesos* contiene procesos, hilos de ejecución, mecanismos de comunicación entre procesos y sincronización.
- La vista de *Implementación* contiene módulos y subsistemas.
- La vista de *Despliegue* contiene los nodos físicos del sistema y las conexiones entre nodos.

4.3.4.2.2 *Diagramas*

Cada componente de proceso implica la construcción de ciertos diagramas o modelos del sistema en desarrollo. Estos diagramas se especifican por medio de la notación UML. El Proceso Unificado Rational, incluye los siguientes diagramas:

- Diagrama de Casos de Uso: especifican la funcionalidad del sistema desde el punto de vista del usuario. Incluye actores, casos de uso y las relaciones entre estos. En la figura 4.3 se observa un diagrama de casos de uso en el que se especifican los casos de uso de un actor Usuario, que puede iniciar los casos de uso: Ingresar Palabras Claves, Solicitar Información y Recomendar al Usuario.

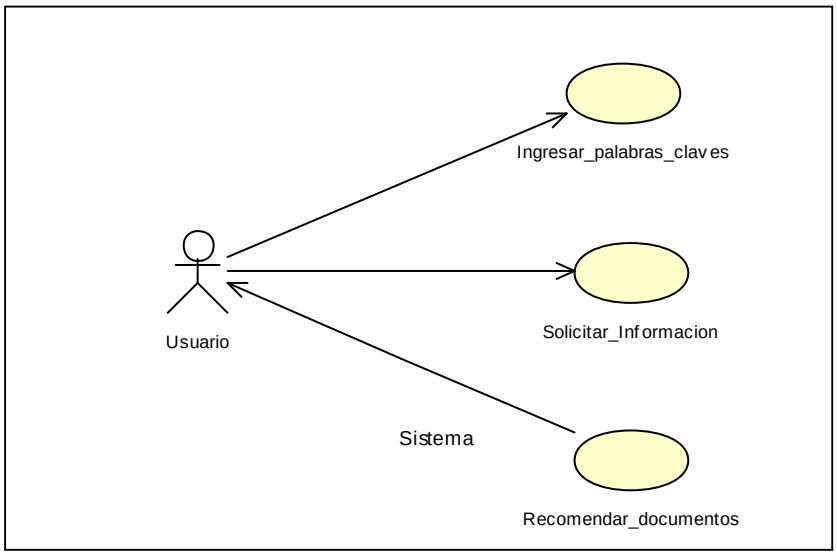

Figura 4.3: Diagrama de Casos de Uso

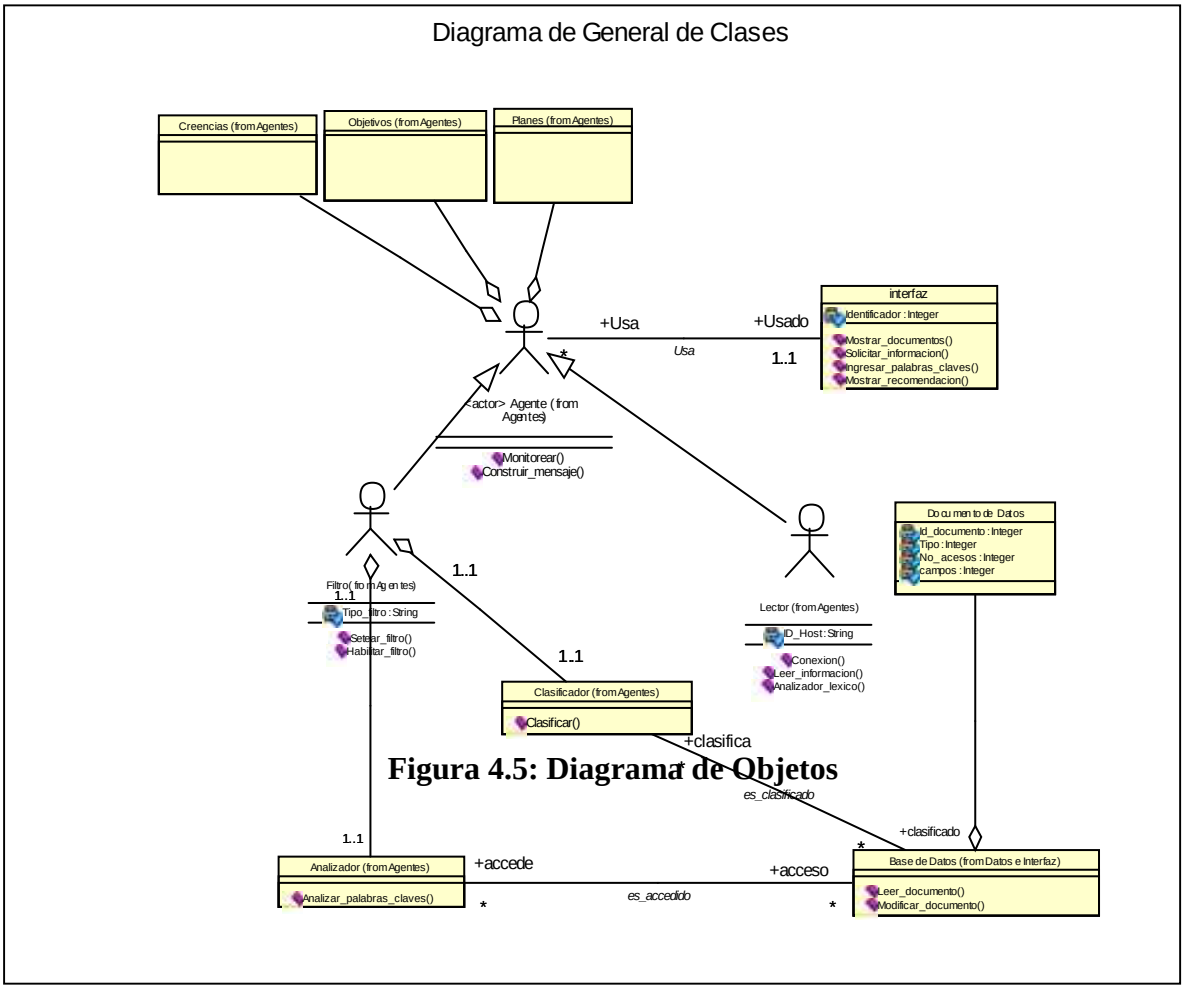

Figura 4.5: Diagrama de Objetos

Figura 4.4: Diagrama de Clases

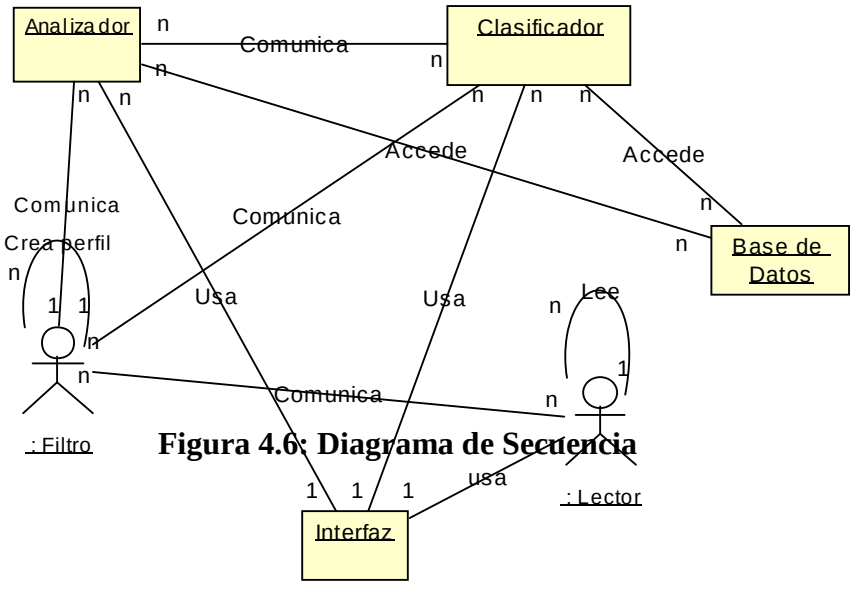

Figura 4.6: Diagrama de Secuencia

Diagrama de Clases y Objetos: especifica la estructura estática del sistema, incluye las entidades estructurales del sistema (clases y objetos) y las relaciones entre los mismos. En el diagrama de clases de la figura 4.4 se observan varias clases como son Base de Datos, Interfaz entre otras, y las relaciones de generalización, agregación y asociación entre ellas. En la figura 4.5 se observa un diagrama de objetos en el se modelan las relaciones semánticas entre los objetos Analizador, Clasificador, Lector, Filtro, Interfaz y Base de Datos por medio de asociaciones, especificando además la cardinalidad.

- Diagrama de Secuencia: especifica la vista dinámica del sistema. Incluye a los objetos del sistema y la interacción entre los mismos por medio del paso de mensajes. Incluye como componente principal al tiempo como variable, y por tanto es orientado al tiempo. El diagrama de la figura 4.6, modela la interacción entre los objetos Lector, Analizador, Servidor Web, por medio del paso de mensajes. En este diagrama (por ser de secuencia) el orden está dada por la variable tiempo (la línea vertical).
- Diagrama de Colaboración: especifica al igual que el diagrama de secuencia la vista dinámica del sistema. La diferencia con el diagrama anterior es que el diagrama de colaboración es orientado a mensajes y por tanto no incluye al tiempo. En figura 4.7, se muestra la colaboración entre los distintos objetos y como se indico antes el componente principal es el paso de mensajes y se puede notar que también es posible modelar el hecho de que un mensaje

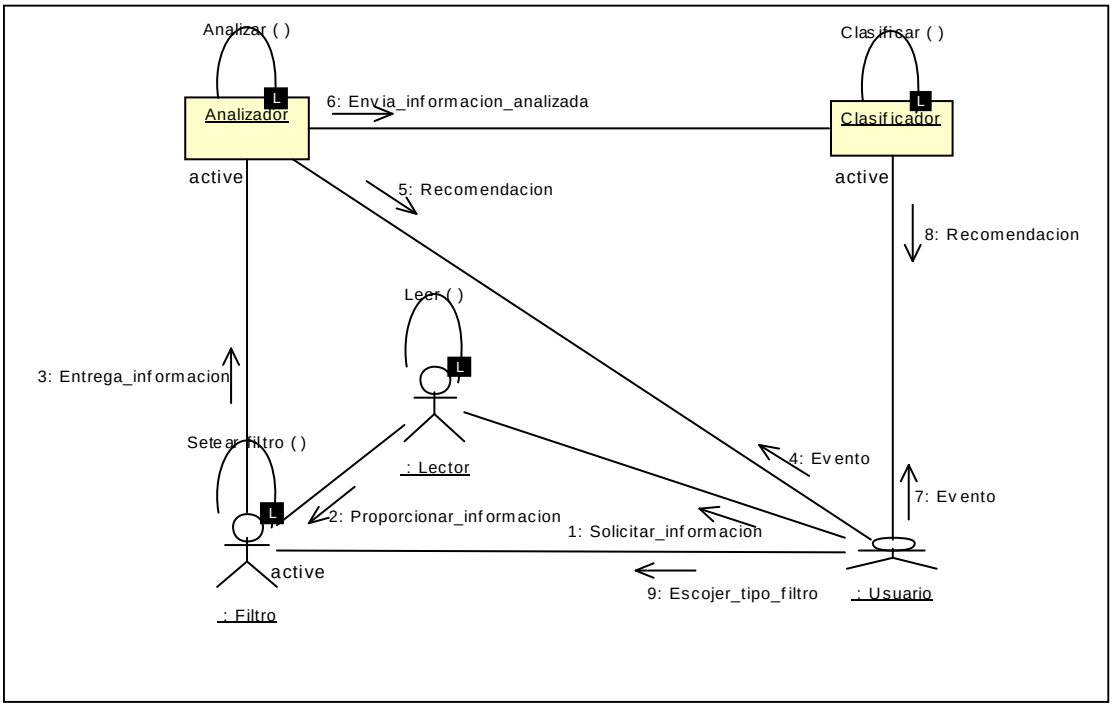

Figura 4.7: Diagrama de Colaboración

requiere una respuesta. Por ejemplo, cuando se envía el mensaje Evento del objeto Usuario al objeto Analizador se devuelve como respuesta al emisor el mensaje Recomendación.

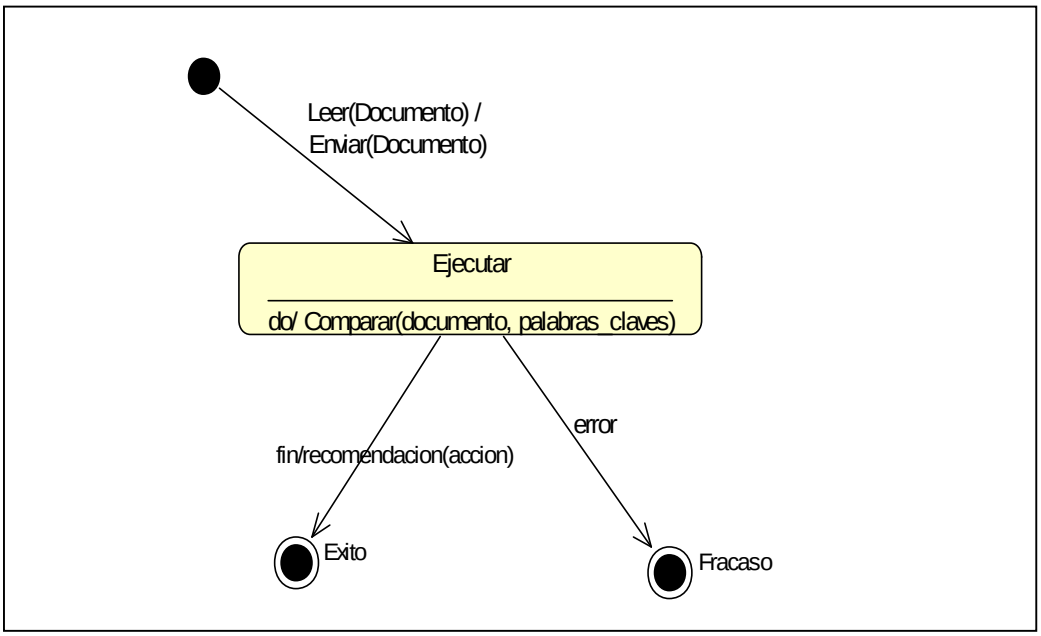

Figura 4.8: Diagrama de Transición de Estados

- Diagrama de transición de estados: al igual que los dos diagramas precedentes, captura el comportamiento dinámico, pero a diferencia está orientado a eventos. El diagrama de estados de la figura 4.8 modela los posibles estados de una clase en la que cada transición se dispara al ocurrir un evento y opcionalmente en algunas se ejecuta además una acción. En cada estado se especifica la acción a ejecutar, por ejemplo en el estado Ejecutar se ejecuta la acción comparar(documento, palabras claves) en el proceso del estado.
- Diagrama de Componentes o de Módulos: captura la estructura física de la implementación, ya que incluye a los módulos que implementan al sistema. En la figura 4.9, se pueden ver los módulos Agente e Interfaz, con la relación de dependencia o uso entre los mismos.

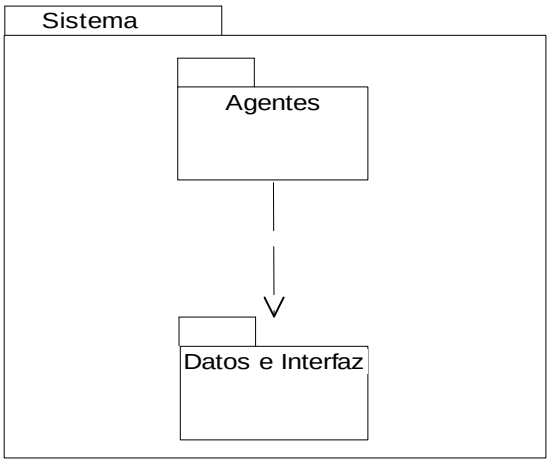

Figura 4.9: Diagrama de Módulos

- Diagrama de Despliegue o de Procesadores: captura la topología del hardware del sistema, además de incluir la asignación de módulos a procesadores. La figura 4.10, muestra un procesador.

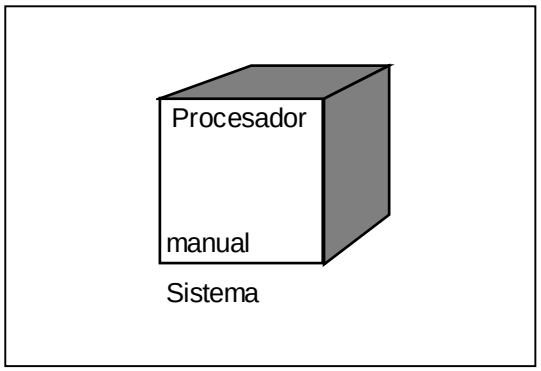

Figura 4.10: Diagrama de Procesadores

4.3.4.2.3 *Relación Diagramas y Componentes de Proceso*

En la tabla 4.1, se observa por cada Componente de Proceso los diagramas que deben ser incluidos. Para indicar el hecho que la inclusión de un diagrama es opcional se marca con una ¼, mientras que cuando es obligatorio se marca con una Ě

Diagramas/Comp. de Proceso	**Requerimientos**	**Análisis**	**Diseño**	**Implementación**	**Prueba**
Casos de Uso	Ě				Ě
Clases		Ě	Ě		Ě
Objetos		Ě	Ě		Ě
Secuencia	Ě	¼	Ě	Ě	Ě
Colaboración		Ě	¼	¼	Ě
Transición de Estados	¼	¼	Ě		Ě
Módulos				Ě	Ě
Procesadores				Ě	Ě

TABLA 4.1: Componentes de Proceso – Diagramas

4.1.3.2 dyOSL Lenguaje Formal de Especificación de Objetos dinámicos

Las metodologías Orientadas a Objetos convencionales (Booch [BOOCH94], OMT [RUMB91], OOSE [JACOB97], Proceso Unificado Rational [KRUCH98], entre otras) son lo suficientemente expresivas como para modelar comportamiento cambiante de objetos que dependen de cambios de sus estados; pero con la limitación de que *estas modificaciones deben ser predefinidas en tiempo de especificación, es decir, antes de la creación de los objetos*.

Como se lo especificó en el capítulo 1, se puede considerar a un agente como un objeto activo, autónomo, concurrente y evolutivo que está equipado con capacidades de razonamiento y conocimiento y que puede tratar con aspectos dinámicos, por ejemplo cambiar su estado así como su comportamiento en forma dinámica.

A fin de captar los efectos de la evolución de un objeto, se debe encontrar modelos semánticos para objetos donde la especificación del comportamiento del mismo pueda ser modificada durante su existencia. Un modelo de este tipo es el dyOSL (Dynamic Object Specification Language - Lenguaje de Especificación de Objetos Dinámicos) propuesto por Saake et al [SAAKE95].

El lenguaje dyOSL utiliza especificación de objetos por medio de lógica temporal utilizando una variante de especificación lógica de objetos (OSL [SERNAD94]).

object creencias_filtrar_agrupar_predictivo
template
attributes **vector**[**n**][**2**]: int ;
/*representa una lista de las palabras claves que deben ser para analizar los documentos.*/
ultimo: int; /*el último espacio ocupado en la lista palabras claves*/
archivo_actual: int; /*identifica el documento analizado */
Agrupar: boolean, **Predicción**: boolean;
events
birth **Inicializar**();
Reset();
Modificar_Creencia(**old**:string, **new**:string);
Agregar_Creencia(**new**: string);
Eliminar_Creencia(**old**:string);
Agregar_calificación(**acción**: int, **calificación**: int);
Agregar_clasificación(**acción**: int, **clasificación**: int);
specification

BaseAxioms=

begin axioms

valuation

[**Inicializar**] **vector**[][]={ },

id_documento=Null, **último**=0,

Agrupar=false, **Predicción**=false;

[**Agregar_calificación**(**acción, calificación**)]

último = **último** + 1,

vector[**último**][**1**]=acción,

vector[**último**][**2**]=calificación;

[**Agregar_clasificación**(**acción, calificación**)]

último = **último** + 1,

vector[**último**][**1**]=acción,

vector[**último**][**2**]=clasificación;

end axioms;

behaviour valuation

[**Inicializar**] **Axioms** = **BaseAxioms**;

[**Reset**] **Axioms** = **BaseAxioms**;

[**Modificar_Creencia**(**x**, **x'**)] **Axioms** = (**Axioms** – {**x**}) U {**x'**};

[**Agregar_Creencia**(**x**)] **Axioms** = **Axioms** U {**x**};

[**Eliminar_Creencia**(**x**)] **Axioms** = **Axioms** – {**x**};

[**Agregar_calificacion**(**x**) and **Agrupar**]

Axioms=Axioms + {[**Agregar_calificacion**(**calificacion**)]

ultimo=**ultimo**+1,

vector[**ultimo**][**1**]=**id_documento,**

vector[**ultimo**][**2**]=**calificacion**;}

[**Agregar_clasificacion**(**x**) and **Prediccion**]

Axioms=Axioms + {[**Agregar_calificacion**(**clasificacion**)]

ultimo=**ultimo**+1,

vector[**ultimo**][**1**]=**id_documento**,

vector[**ultimo**][**2**]=**clasificacion**;}

end object creencias_filtrar_agrupar_predictivo

Figura 4.11: Especificación de un objeto evolutivo en dyOSL

En el lenguaje dyOSL un objeto se especifica por medio de plantillas similares a la de la figura 4.11. La especificación del comportamiento de un objeto consiste de dos niveles: una especificación del comportamiento usual del objeto en términos de eventos y atributos, y una de *modificaciones* a esta especificación base. Con este propósito, se introduce un atributo implícito **Axioms** que contiene los axiomas actuales de la especificación, más los atributos. Como se puede notar en la figura 4.11, se tiene un **BaseAxioms** que representa el estado inicial del objeto. Este incluye los valores iniciales de los atributos y además provee la definición usual de las operaciones definidas como métodos del objeto. El comportamiento dinámico se modela al definir la sección de comportamiento, en donde el atributo **Axioms** (implícito), al cambiar de contenido puede implicar (si está especificado) modificación de los atributos y/o de la definición de las operaciones que implementa el objeto.

4.1.4 METODOLOGÍAS ORIENTADAS A AGENTES

En esta sección se presenta una descripción sintética de las dos metodologías Orientadas a Agentes seleccionadas para el trabajo de investigación.

4.1.4.1 Técnica de Modelado de Agentes para Sistemas de agentes BDI

Esta metodología es una extensión de las técnicas de modelado Orientado a Objetos aplicada a sistemas de agentes BDI [KINNY96].

Al especificar un sistema basado en agentes utilizando esta metodología, se adopta un conjunto de diagramas que operan en dos niveles distintos de abstracción. El primer nivel es *la vista externa*, en ésta, el sistema se descompone en agentes, modelados como objetos complejos caracterizados por sus propósitos, sus responsabilidades, los servicios que realizan, la información que requieren y mantienen, y su interacción con el exterior. En el segundo nivel, conocido como *vista interna*, por cada agente se modelan los elementos de la arquitectura particular del agente, que son: sus creencias, objetivos y planes.

4.3.4.2.1 *Vista externa*

Los detalles de la vista externa se capturan en dos modelos:

1. El *Modelo de Agentes*, que describe la relación jerárquica entre las diferentes clases de agentes concretos y abstractos, e identifica las instancias de los agentes, su multiplicidad, y el momento en que inician su existencia. Este modelo incluye dos componentes:

- Un Modelo de Clases de Agentes, que es un conjunto de diagramas de clases que definen agentes concretos y abstractos; y captura las relaciones de herencia y agregación entre ellos. En la figura 4.12, se tiene un diagrama de Clases de Agentes, con los agentes: Filtro, Lector y Usuario. El agente Asistente es una agregación de un agente Analizador y Clasificador.
- Un Modelo de Instancias de Agentes, que es un conjunto de diagramas de instancias que identifican las instancias de los agentes definidos en el modelo de clases de agentes. En la figura 4.13, se especifica un diagrama de instancias de agentes, en el cual los agentes Lector se instancia en tiempo de compilación, esto se modela mediante el símbolo S que aparece en la esquina superior izquierda de la clase. Mientras que Filtro se instancia en tiempo de ejecución. Las instancias se modelan mediante rectángulos con bordes curvos, unidos a las clases a través de flechas punteadas. Para indicar que la instanciación es múltiple se indica con flechas cuyo origen es un círculo pintado.

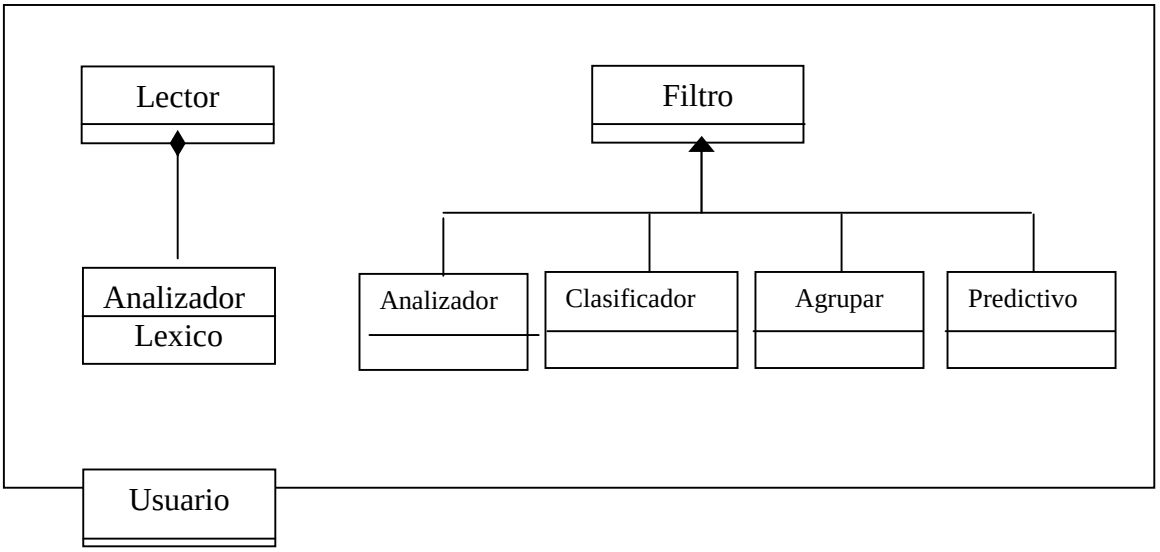

Figura 4.12: Diagrama de Clases de Agentes

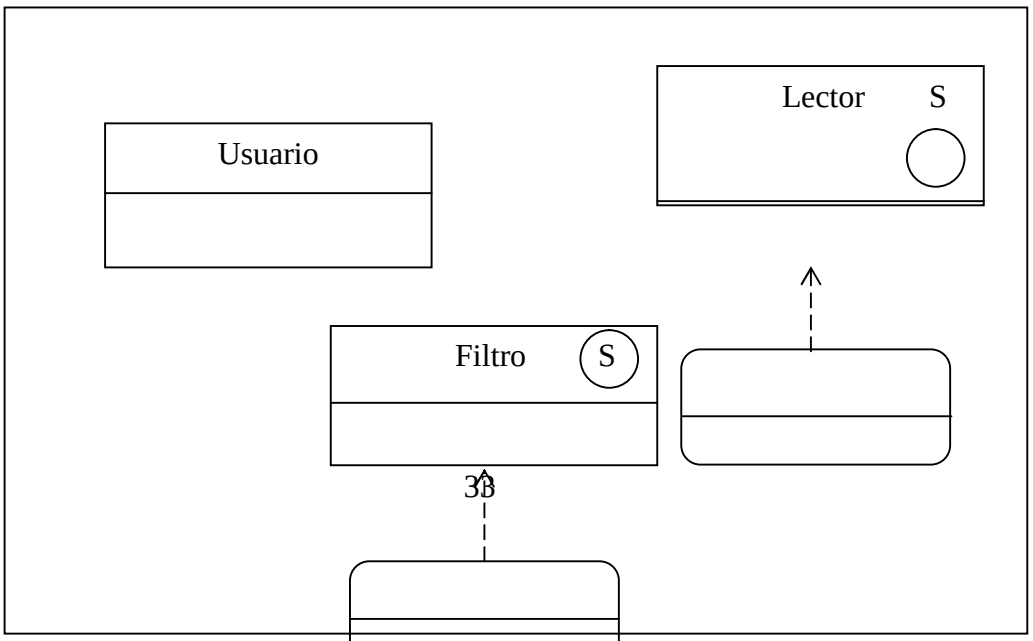

Figura 4.13: Diagrama de Instancias de Agentes

2. El *Modelo de Interacción*, que describe las responsabilidades de una clase de agente, los servicios que provee, las interacciones asociadas y las relaciones de control entre las clases de agentes. En la figura 4.14, se tiene una interacción para un agente Lector, el paso de mensajes se especifica en el lenguaje KQML [FININ92].

Agente Lector:

Responsabilidad: Comunicarse con otro agente

Servicios:

1. Monitorear el ambiente para captar los mensajes de otro agente

Mensaje KQML

monitor

:content *documento*
:language *ASCII*
:ontology *requerimiento*
:reply-with *NIL*
:force *permanent*
:sender *Lector*
:receiver: *Filtro*

Figura 4.14: Modelo de Interacción

4.3.4.2.2 *Vista Interna*

La arquitectura BDI provee una visión fuerte de la noción de agentes; según la cual, los agentes poseen ciertas actitudes mentales: Creencias, Deseos e Intenciones; que representan, respectivamente,

los estados de información, motivación y deliberación del agente. Esto se captura para cada clase de agente, en los siguientes modelos:

1. Un *Modelo de Creencias*, que describe la información acerca del ambiente y el estado interno que un agente de esa clase puede poseer, y las acciones que puede realizar. Un modelo de creencias consiste de un conjunto de creencias y uno o más estados de creencias. Como se muestra en la figura 4.15, un conjunto de creencias se especifica como un conjunto de diagramas de objetos que denotan el dominio de las creencias de una clase de agentes.
2. Un *Modelo de Objetivos* que describe los objetivos que un agente puede adoptar, y los eventos a los cuales puede responder. Consiste de un conjunto de objetivos que especifica el dominio de eventos y objetivos, y uno o más estados de objetivos utilizados para especificar el estado mental del agente. En la figura 4.16, se observa el modelo de Objetivos de un agente Filtro, allí se puede notar que los objetivos pueden ser de tres tipos. El símbolo !, indica objetivo a lograr; el símbolo ?, indica objetivo a verificar; mientras que el símbolo $, denota objetivo a determinar.
3. Un *Modelo de Planes* que describe los planes que un agente puede emplear para lograr sus objetivos. Consiste de un conjunto de planes que describe las propiedades y las estructura de control de los planes individuales. En la figura 4.17, se tiene un plan, en el que se puede apreciar que un plan es un diagrama de transición de estados extendido con la noción de eventos y acciones.

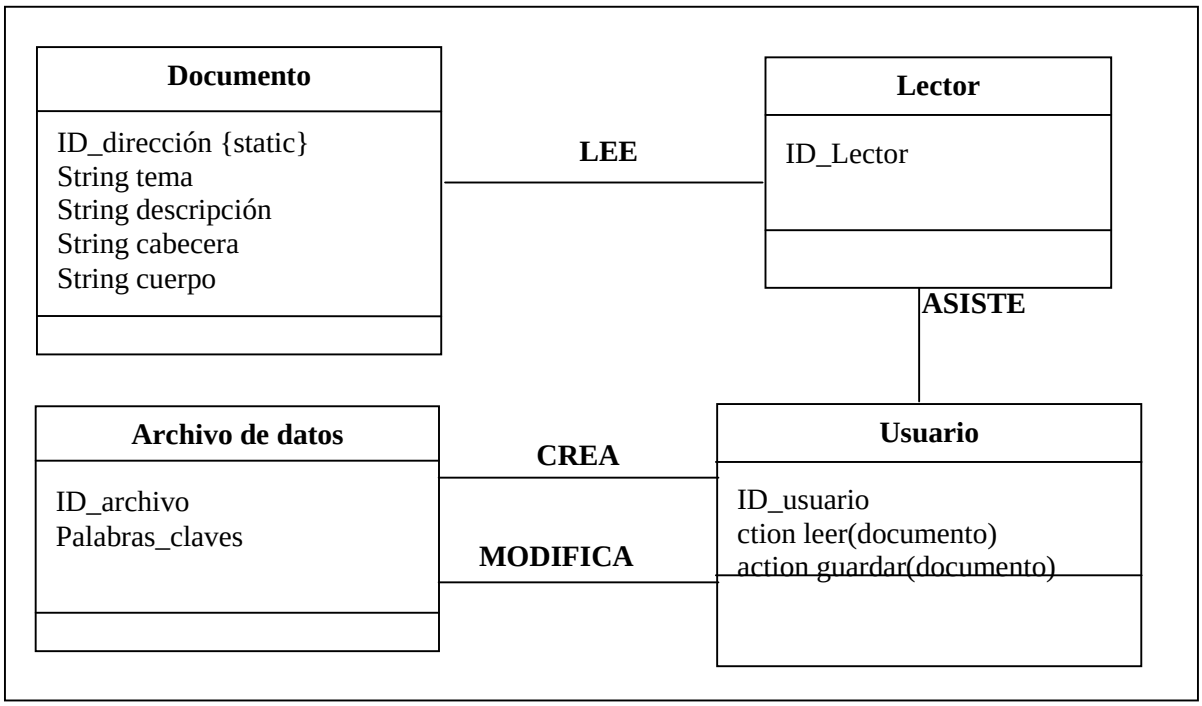

Figura 4.15: Diagrama de Creencias

Agente Filtro
compara(Documento, Base de Datos)?
analizar(Documento)!
modifica(Filtro, Base de Datos)$
clasifica(Filtro, Documento)!
califica(Filtro, Documento)!
recomienda(Filtro, Usuario)$

Figura 4.16: Conjunto de Objetivos

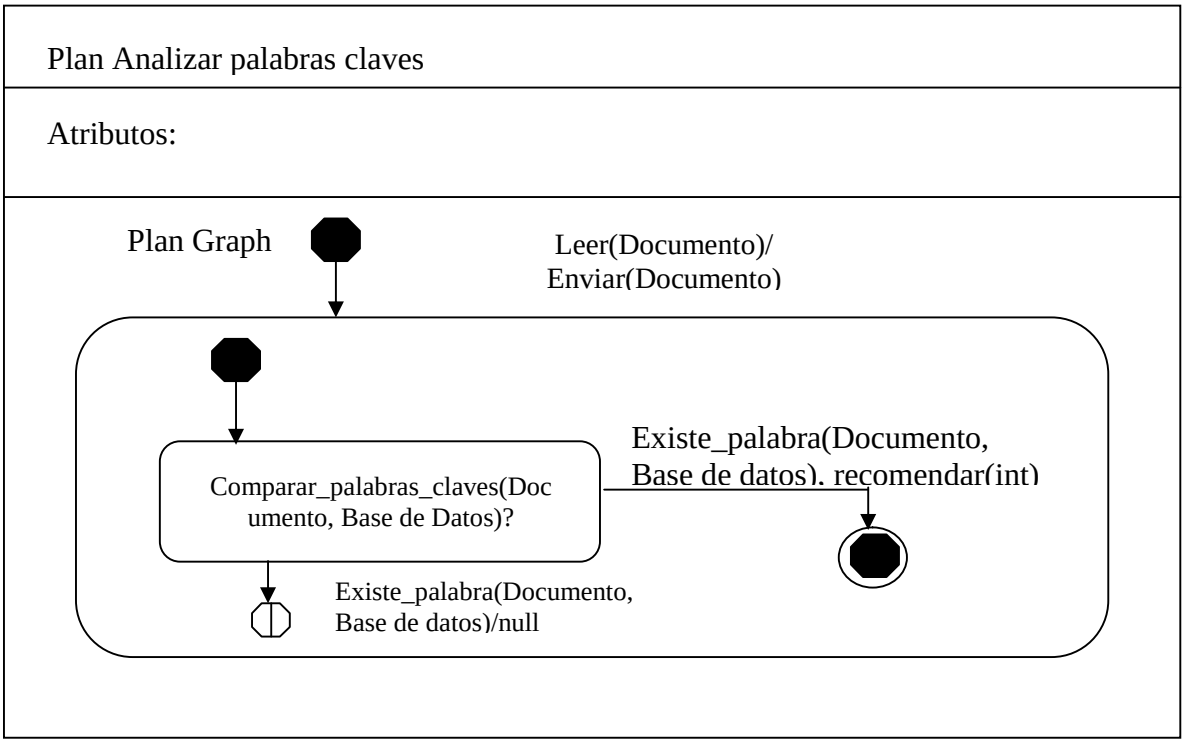

Figura 4.17: Diagrama de Planes

4.1.4.2 Metodología MAS-CommonKADS

La Metodología MAS-CommonKADS [IGLESI97], pertenece al grupo de metodologías Orientadas a Agentes que son una extensión de metodologías de Ingeniería del Conocimiento.

Al igual que el Proceso Unificado Rational, el modelo de ciclo de vida propuesto por esta metodología para el desarrollo de sistemas es el modelo en espiral dirigido por riesgos [BOEHM88].

MAS-CommonKADS propone los siguientes modelos para el desarrollo de sistemas multiagente:

- *Modelo de Agente (AM):* específica las características de un agente: sus capacidades de razonamiento, habilidades, servicios, sensores, efectores, grupos de agentes a los que pertenece y clase de agente. Un agente puede ser un agente humano, software, o cualquier entidad capaz de emplear un lenguaje de comunicación de agentes.
- *Modelo de Organización (OM):* es una herramienta para analizar la organización humana en que el sistema multiagente va a ser introducido y para describir la organización de los agentes software y su relación con el entorno.
- *Modelo de Tareas (TM):* describe las tareas que los agentes pueden realizar: los objetivos de cada tarea, su descomposición, los ingredientes y los métodos de resolución de problemas para resolver cada objetivo.
- *Modelo de la Experiencia (EM):* describe el conocimiento necesitado por los agentes para alcanzar sus objetivos. Sigue la descomposición de *CommonKADS* y reutiliza las bibliotecas de tareas genéricas.
- *Modelo de Comunicación (CM):* describe las interacciones entre un agente humano y un agente software. Se centra en la consideración de factores humanos para dicha interacción. Aún no se ha sido desarrollado en la metodología.
- *Modelo de Coordinación (CoM):* describe las interacciones entre agentes software.
- *Modelo de Diseño (DM):* mientras que los otros cinco modelos tratan del análisis del sistema multiagente, este modelo se utiliza para describir la arquitectura y el diseño del sistema multiagente como paso previo a su implementación.

4.3.4.2.1 *Conceptualización*

Antes de iniciar la especificación de los modelos descritos más arriba, la metodología MAS-CommonKADS incluye una etapa preliminar: la fase de conceptuación, empleando análisis centrado en el usuario mediante la técnica de casos de uso. Esta técnica permite definir los usos que dan los distintos usuarios al sistema. La notación propuesta para la fase conceptualización es tanto textual como gráfica. La notación textual consiste en plantillas textuales que especifican los casos de uso y los actores, mientras que la notación gráfica sigue una extensión de la notación original de Jacobson

[JACOBS97] (ver figura 4.3) para mostrar también las interacciones entre el sistema y los actores mediante diagramas de intercambio de mensajes (ver figura 4.6).

4.3.4.2.2 *Modelo de Agentes*

El objetivo de este modelo es la descripción de los agentes software y humanos que operan en el sistema y sirve como punto de partida del resto de modelos de análisis de *MAS-CommonKAD*S. El modelo de agente es una herramienta para analizar el sistema desde una perspectiva orientada a agentes. El desarrollo del modelo de agente consta de una fase de identificación de agentes, soportada por diferentes técnicas, que permite modelar el problema con agentes, y una fase de descripción de los mismos, en la que se describe con más detalle cuáles son las tareas encomendadas a los agentes y sus características. El resultado de esta etapa es un conjunto de plantillas de agentes similares al de la figura 4.18, en la que se describe las cualidades del agente. Además un conjunto de objetivos identificados por cada agente, los objetivos se describen por medio de plantillas de objetivos como se muestra en la figura 4.19

Agente *Filtro*

tipo

agente software inteligente estacionario.

descripción

Este agente se encarga de filtrar la información, proporcionada por el Lector en su tarea de leer documento, identificando palabras claves, agrupamiento y predicción y recomendar los documentos al usuario. Para ello debe monitorear las acciones del usuario y en forma autónoma asistirlo.

este agente se

Capacidad-razonamiento

experiencia

Conocimiento de prácticas de clasificación, conocimiento del repositorio de clasificaciones (repositorio que contiene todas los datos generadas por la herramienta).

comunicación

Interacción con el Usuario, interacción con el agente Lector.

coordinación

En cada momento el agente Lector se comunica con el agente Filtro.

Capacidad-general

habilidades

monitorear las acciones del Usuario

lenguaje-com

Direcciones web para los sensores de entrada, y lenguaje natural para los efectores (sensores de salida).

Restricción

normas

El agente *Lector* del actor *Usuario* es el encargado de comunicarse con el agente *Filtro*. El agente *Filtro* modifica el repositorio de datos que existe a través del actor Usuario.

preferencias

Política del buzón (FIFO)

permisos

El agente *Filtro* puede modificar el repositorio de datos correspondiente solo al *Usuario* al que asiste..

Figura 4.18: Modelo de Agente: plantilla de Agente

Objetivo Monitorear el ambiente para detectar eventos del Usuario

tipo

Objetivo persistente

parámetros-entrada

Documento

parámetros-salida

Parámetros de calificación y clasificación.

condición-activación

Inicio de la aplicación.

condición-finalización

Fin de la aplicación

lenguaje rep. conocimiento

Documento para la entrada, parámetros de calificación y/o clasificación interna para la salida.

descripción

Este objetivo permite el flujo de eventos en todo el sistema multiagente, ya que provee de entrada a todas las tareas..

Figura 4.19: Modelo de Agentes: plantilla de objetivos

4.3.4.2.3 *Modelo de Organización*

Los fines del modelo de organización son:

(1) Comprender la organización en que se va a introducir un sistema multiagente y analizar la viabilidad del mismo;
(2) Describir las relaciones de la sociedad multiagente.

En la figura 4.20 se tiene un ejemplo de un diagrama de organización, en el que se describe las relaciones de la sociedad multiagente, además de la relación de los agentes (cajas con los bordes superiores redondeados) con las entidades estáticas del entorno (cajas convencionales para representar clases en una notación Orientada a Objetos) como son Usuario, Interfaz, etc. También se pueden ver relaciones de asociación entre las entidades del sistema.

4.3.4.2.4 *Modelo de Tareas*

El modelo de tareas de MAS-CommonKADS, da una visión de alto nivel de las tareas de la organización que se desea modificar y de sus objetivos. Cada ejemplar del modelo de tareas está relacionado con un objetivo asignado a uno o varios agentes. Las tareas que requieren "experiencia" son desarrolladas en el modelo de la experiencia, mientras que las que requieran transferencia de información son desarrolladas en los modelos de comunicación y coordinación.
En la figura 4.21 se tiene un ejemplar de una tarea especificado utilizando la plantilla para tareas, como se puede ver, primero se especifica el objetivo asociado, luego se provee una pequeña descripción de la tarea, posteriormente se detallan la entrada, la salida, las condiciones de ejecución y la frecuencia. En la figura se observa también un apartado correspondiente a super- y sub- tarea, esto se debe a que la identificación de las distintas tareas resulta de una descomposición top-down de tareas más generales en tareas más específicas.

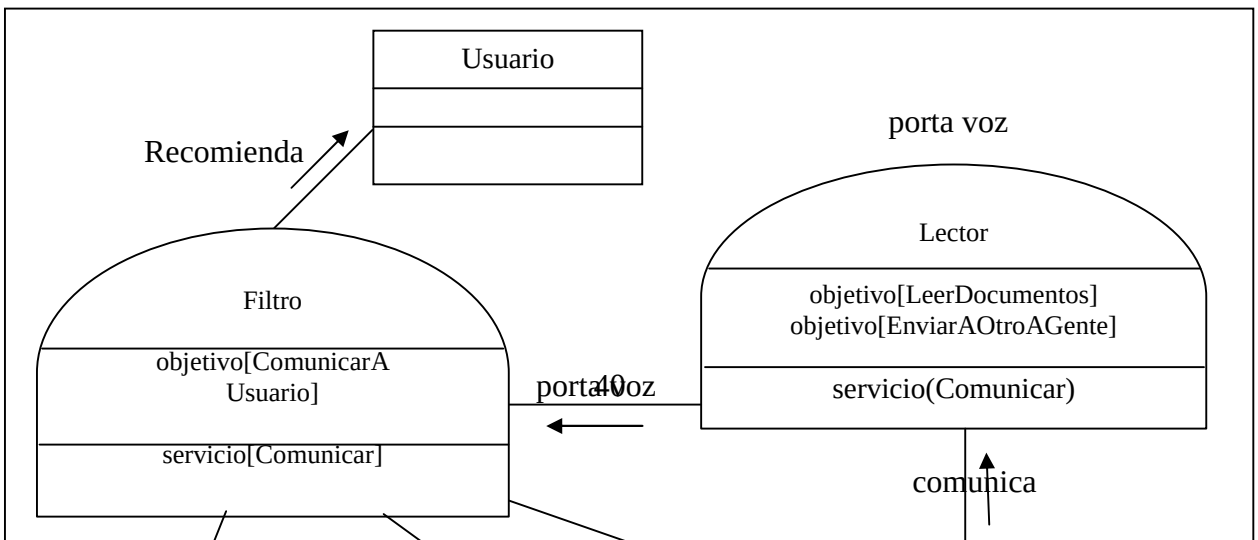

Figura 4.20: Diagrama de Organización, Organización Multiagente

4.3.4.2.5 *Modelo de la Experiencia*

MAS-CommonKADS desarrolla en el modelo de la experiencia las tareas que requieren conocimiento para ser llevadas a cabo y que permitirán caracterizar al agente como un sistema basado en conocimiento. Para ello se describe la ontología o esquema del modelo, es decir, los principales conceptos del dominio y sus relaciones. Como se ve en la figura 4.22, se especifican los conceptos del dominio por medio de plantillas de concepto.

Tarea *ConsultarBaseDeDatos*

objetivo

Encontrar en la Base de datos posibles recomendaciones.

descripción

En esta tarea se buscan posibles recomendaciones, en la base de datos sobre palabras claves, agrupamientos y predicción para un documento especificado por el actor Usuario.

entrada

Documento

salida

Parámetros de calificación y clasificación.

precondición

Documento recibido

supertarea

IdearRecomendación

subtareas

ninguna

frecuencia

Por cada elemento de entrada

Figura 4.21: Modelo de Tareas: plantilla de tarea

Concepto Usuario

descripción

Representa a un usuario de la aplicación

propiedades

Identificador: cadena;

Figura 4.22: Modelo de Experiencia: plantilla de concepto de dominio

4.3.4.2.6 *Modelo de la Coordinación*

Presenta un modelo para describir las interacciones entre los agentes de una arquitectura multiagente. El modelo se estructura en torno a las conversaciones entre agentes. El modelo ofrece diversas técnicas y notaciones para la identificación y descripción de las conversaciones entre los agentes, las interacciones asociadas, y los objetivos que se pretenden con dichas conversaciones. El desarrollo del modelo combina técnicas de metodologías orientadas a objetos con lenguajes de descripción formal de ingeniería de protocolos.

En la figura 4.23 se presenta una plantilla de una comunicación entre tres agentes (Filtro, Lector, Analizador, Clasificador), por cada conversación se definen intervenciones que tiene por objetivo determinar los mensajes intercambiados entre los agentes, en la figura 4.24 se puede ver un ejemplo de una intervención, la notación es parecida a los diagramas de secuencia de la notación UML.

Conversación *Comunicar al Usuario*

tipo

Comunicar-información

objetivo

Comunicar al usuario de las recomendaciones provenientes del agente Filtro.

agentes

Lector, Filtro, Analizador, Clasificador

iniciador

Filtro

servicio

Comunicar

descripción

El agente Filtro tiene por objetivo analizar los documentos recibidos para luego comunicar sus recomendaciones al usuario.

precondición

Recomendación recibida

postcondición

Recomendación comunicada

Figura 4.23: Modelo de la Coordinación: plantilla de conversación

4.3.4.2.7 *Modelo de Diseño*

Este modelo se ha basado en concebir un sistema multiagente como una red que ofrece funcionalidades a los agentes (modelo de red) y una serie de agentes que operan en la red. En este modelo se recogen las especificaciones del resto de modelos y se decide cómo deben llevarse a cabo.

En la figura 4.25, se puede ver una plantilla de sistema agente, en el que se puede especificar la arquitectura del agente a implementar y también en el caso de que tenga algún subsistema. En al figura 4.26, se puede ver una plantilla de plataforma en la que se especifica todos los datos de implementación: lenguaje, plataforma de ejecución, usuario, etc.

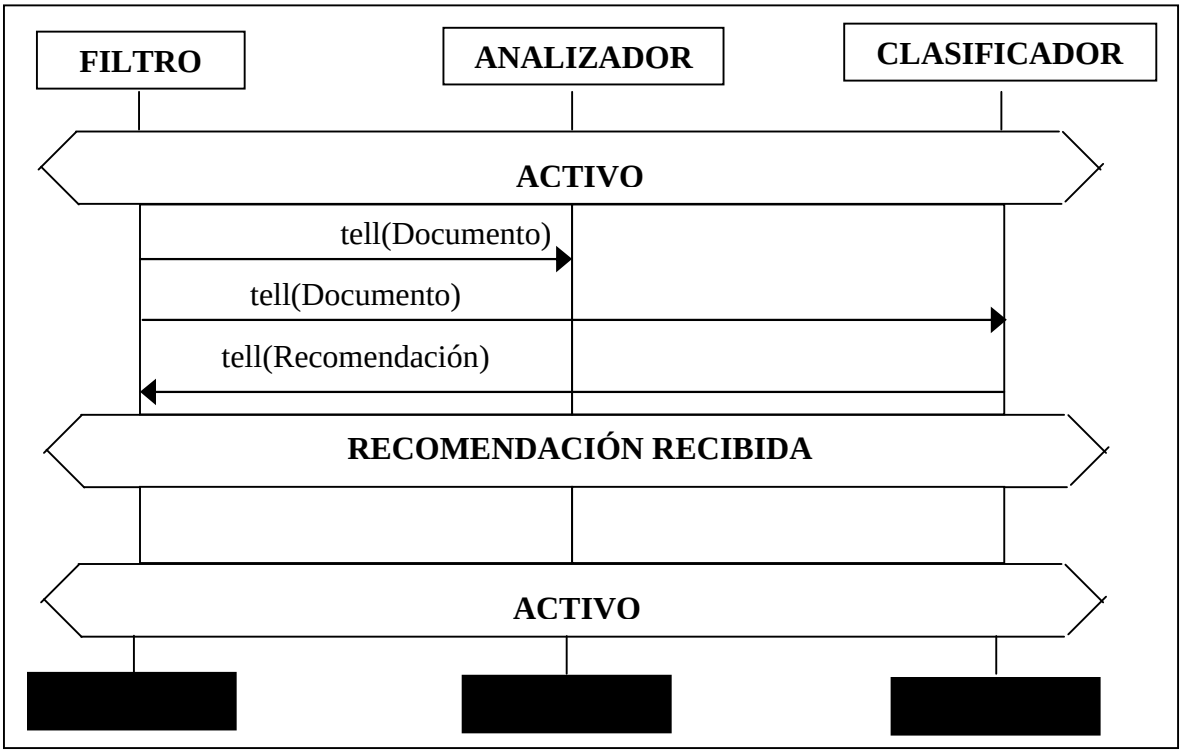

Figura 4.24: Modelo de Coordinación, Intervención

Sistema-Agente *Filtro*

arquitectura

arquitectura híbrida

tiene-subsistema

ninguno

Figura 4.25: Modelo de Diseño, plantilla de Sistema Agente

Plataforma *Sistema-Multiagente*

descripción

no se utiliza plataforma multiagente predefinida.

usa-lenguaje

Java

hardware-requerido

PC con Windows95-98.

software-requerido

Java (j2sdk-1_4_2_03-windows-i586-p.exe),
j2re1_3_0-win.exe

usuario

Usuario

Figura 4.26: Modelo de Diseño, plantilla de Plataforma

4.2 DESARROLLO DE UN AGENTE

En esta sección se presenta una descripción del sistema basado en agentes utilizado en el trabajo de investigación, además se realiza un análisis crítico del proceso de construcción de sistemas basados en agentes aplicando distintas metodologías, así como algunas consideraciones acerca de los resultados obtenidos. El análisis se estructura desde dos puntos de vistas: uno general o global abarcando todo el ciclo de vida de un sistema software; y el otro más puntual para cada etapa de desarrollo del sistema. Después se presenta un resumen comparativo de las metodologías estudiadas. Para finalmente, se esboza una propuesta inicial de ampliación de la metodología Proceso Unificado Rational para sistemas basados en agentes.

El crecimiento del World Wide Web y la naturaleza dinámica de los recursos disponibles en Internet constituyen un reto para los organizadores de la información y han hecho necesaria la creación de herramientas que proporcionen acceso a los millones de documentos que existen en formato electrónico. En efecto, una página web puede ser movida de un sitio a otro dentro de un mismo lugar, desplazada a otro diferente o, puede desaparecer por completo; su contenido puede ser modificado con gran rapidez; puede cambiar la persona responsable de la página, o la institución auspiciadora. A mas de esto se pueden agregar nuevos métodos de acceso diferentes a los originales. Por medio de las herramientas de búsqueda existentes se ha pretendido dar solución a estas variables. Dichas herramientas utilizan métodos automáticos para identificar los recursos de Internet. De hecho, existen muchos programas que “navegan” automáticamente a través de la Web, buscando enlaces, recuperando documentos, indexándolos y creando bases de datos. Estos sistemas recuperan gran cantidad de documentos, pero su precisión es muy baja. La causa principal no es que los métodos automáticos utilizados describan inadecuadamente los recursos electrónicos de Internet, sino que los mismos documentos HTML carecen de información suficiente para describir el recurso adecuadamente. Esto arroja como resultado mucha “basura” en la recuperación de información.

Internet, se dice, crece de una manera exponencial, se asume que será cada vez más difícil navegar y, por lo tanto, localizar información relevante.

Conciente del crecimiento de Internet y de las desventajas que ofrecen los procedimientos, métodos y técnicas que actualmente existen para describir la información; bibliotecarios, informáticos y diseñadores de software, entre otros profesionales, ya han planteado la necesidad de crear descripciones y catálogos que logren identificar los recursos electrónicos en Internet de una manera eficaz que permita una búsqueda y localización más efectiva. Así, se debe reconsiderar la manera en que este tipo de recursos pueden ser representados y organizados de una mejor forma.

La necesidad de organizar la información de Internet ha llevado a emplear esfuerzos por controlar la enorme diversidad de información. Entre estos esfuerzos se destacan principalmente dos: los motores de búsqueda, por un lado, que son utilizados como una herramienta de proyectos patrocinados por instituciones que tienen como objetivo agrupar la información de recursos en Internet, y por otro lado, el que ha resultado del "enfoque metadata".

Los primeros, como ya se sabe, son "una herramienta que permite acceder mediante palabras claves, a los recursos de Internet indexados en su base de datos". Previa a esta indexación, se realiza una selección y una evaluación de cada uno de los recursos, con criterios establecidos por cada proyecto.

El enfoque metadata se utiliza para designar el conjunto de elementos informativos que pueden utilizarse para describir y representar objetos de información.
Encontrar una solución al problema de la organización bibliográfica en Internet, se ha convertido en materia de estudio de muchas instituciones, entre las que se destaca los esfuerzos del Online Computer Library Center (OCLC); ya en 1996 Hsieh-Yee puntualizaba que aun cuando era deseable contar con un sistema de información que organice todo lo que existe en Internet, en ese momento tal sistema no era factible ni podría ser efectivo por las siguientes razones:

1. Existen demasiados recursos en Internet
2. La calidad de muchos de esos recursos es cuestionable
3. Muchos recursos son de naturaleza efímera y pueden ser de escaso valor para los usuarios

Entre los primeros intentos encaminados implementar un ordenamiento a Internet, figuran ***The Aarhus Clearinghouse***. Este proporciona un acceso central temático, el cual identifica, describe, y evalúa información de recursos en Internet. Esta actividad es regida por una serie de criterios como son: el nivel de descripción del recurso, el diseño, la organización de sus índices, así como la metainformación que proporciona. Otro es ***CyberStacks***, el cual se define a sí mismo como "Colección centralizada, integrada y unificada de recursos de Internet". Este es un sitio experimental que ordena y

selecciona información de recursos de Internet en las áreas de ciencia y tecnología, de acuerdo a la clasificación de la Biblioteca del Congreso de los Estados Unidos, utilizando criterios como son autoridad, exactitud, claridad, originalidad, etc. Otro proyecto, el ***FiltroInfo***, presenta una serie de criterios para seleccionar y evaluar recursos en Internet, como son, autoridad, contenido, organización, aceptación o uso general, motores de búsqueda y diseño de gráficos entre otros.

Es importante puntualizar que las iniciativas con enfoque metadata se plantean como objetivo principal: que cada uno de los recursos sea "catalogado", para lo cual proponen un cierto número de elementos que describirían al recurso, y que entonces, se volverían susceptibles de ser claves de localización. El enfoque de los motores de búsqueda deja a un lado la propuestas de organización en su estructura de metadatos, su principio de agrupamiento, se basa en establecer una serie de elementos que debe reunir cada unos de los recursos seleccionados de Internet, con el objeto de evaluar su posible integración a su base de datos; esta característica garantiza la calidad de la información. Estas mismas iniciativas son limitadas porque algunas solo se enfocan a ciertas áreas específicas y otras no hacen notoria su participación.

4.3 DESCRIPCIÓN DEL PROTOTIPO

La aplicación *FiltroInfo* desarrollada en este capítulo provee la funcionalidad básica de un filtro de información de propósito general; trata sobre un agente que lee páginas Web y un agente inteligente que ayuda al usuario para filtrar información. El usuario puede especificar un perfil de palabras claves y proveer rangos explícitos para cada pagina web; existen tres métodos alternativos para filtrar los artículos: palabras claves, agrupamientos y modelación predictiva, basada en la clasificación por parte del usuario.

Los agentes son creados para recoger información desde páginas web de Internet. El **Agente Lector de URL** lee los datos desde una dirección específica y descarga la información. Después, cada página web es calificada de acuerdo a una lista de palabras claves y/o por medio del uso de dos redes neuronales. El usuario puede calificar la información por medio del uso de cinco niveles. Los temas de cada pagina web son desplegados en una tabla de control. El usuario puede mostrar el contenido de las páginas web seleccionándolas de la tabla.

Una de las metas de esta aplicación es mostrar como una aplicación basada en agentes puede ser usada para ordenar inteligentemente páginas web de acuerdo a su contenido.

4.3.1 ARQUITECTURA DEL SISTEMA MULTIAGENTE

Luego de tener una vista global de la arquitectura del sistema multiagente se pasa a analizar en detalle la arquitectura interna de cada agente. Básicamente, como ya se especificó en la sección anterior, la sociedad multiagente está formada por dos tipos de agentes distinguibles: un Filtro de información y un Lector con funciones bien definidas.

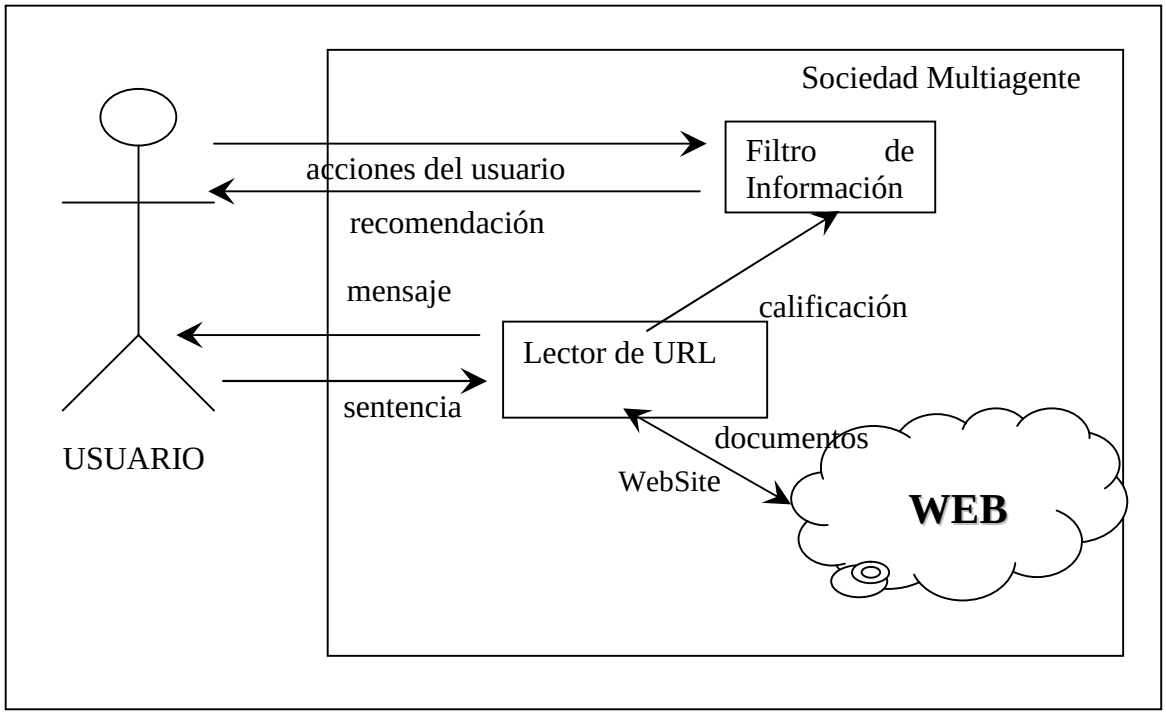

Figura 4.27: Estructura Organizativa de la sociedad Multiagente.

Retomando la definición de agente adoptada [WOOLDR95a] (ver Capítulo 2, Sección 2.1.2) se puede evidenciar que ambos agentes, presentan características de:

- Reactividad: en el caso del Filtro de información al tener que clasificar y filtrar cada entrada. En el caso del Lector al tener que recuperar información, por cada dirección de Páginas Web (URL), por cada sentencia.
- Deliberación: el Filtro de información califica y filtra páginas de acuerdo a una lista de palabras clave y/o por medio del uso de redes neuronales. El Lector analiza el contenido de cada página.
- Sociabilidad: al comunicarse con otros agentes y con el usuario.
- Autonomía: al decidir comunicar o no al usuario un mensaje.
- Nociones mentales: el agente Filtro precisa mantener cierto conocimiento referente al usuario (referencias, palabras claves, preferencias) de modo que pueda ofrecer recomendaciones,

cuestiones de interés, lograr aceptación máxima por parte del usuario a las distintas recomendaciones, etc.

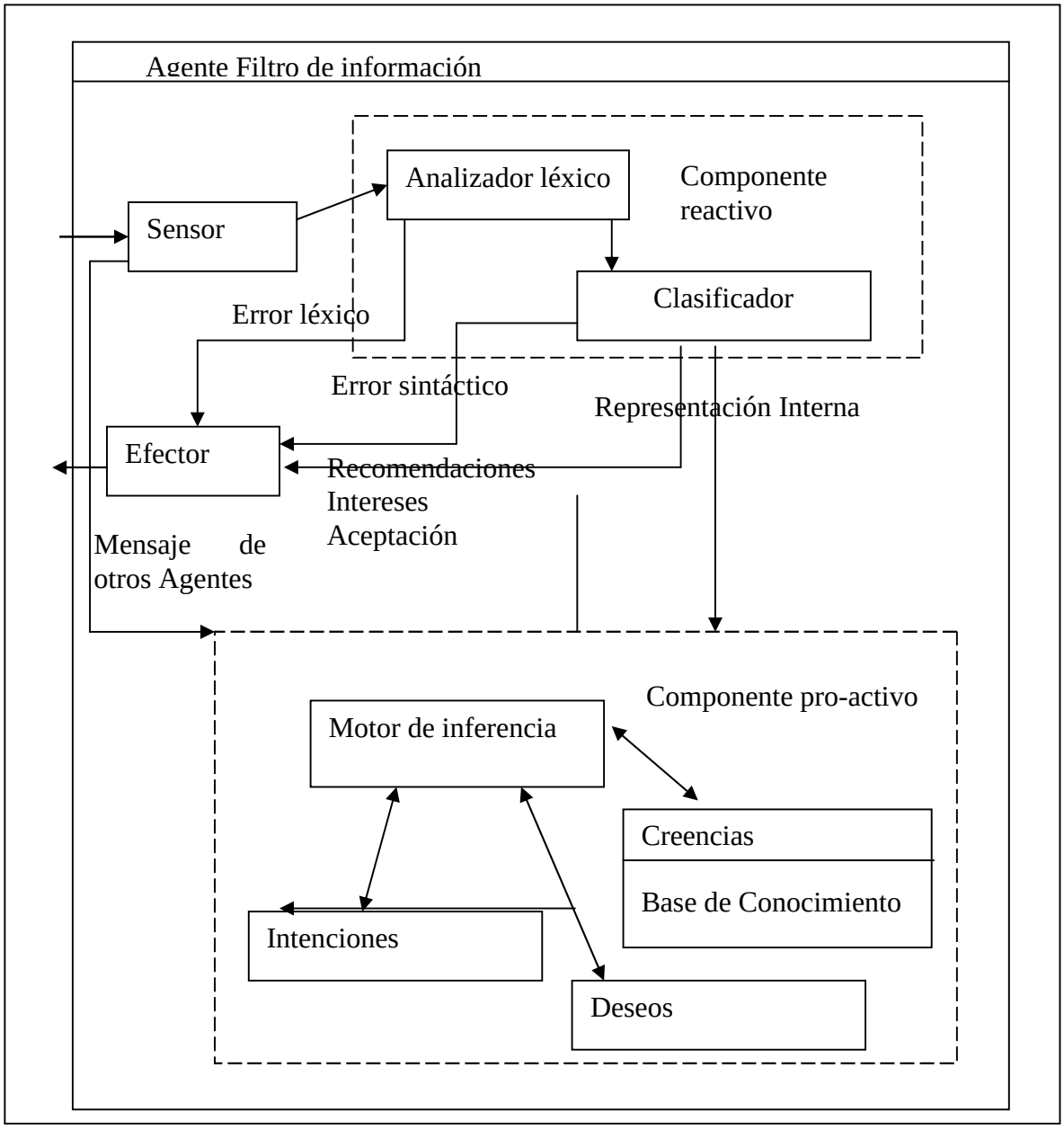

Figura 4.28. Arquitectura del Filtro de información

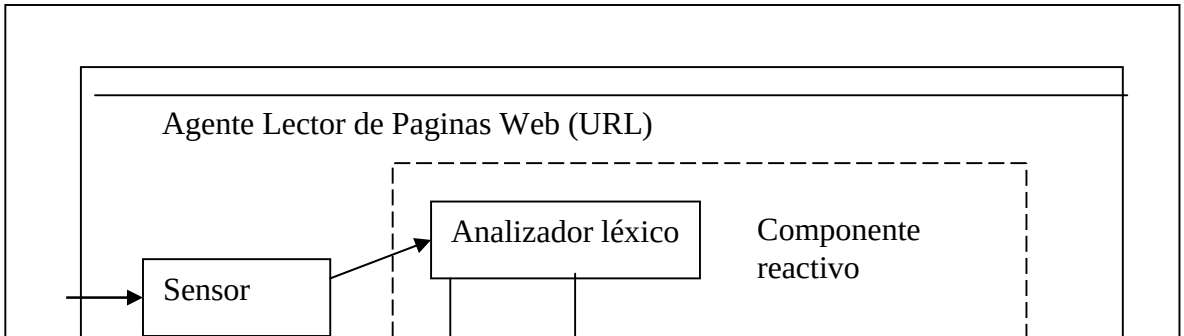

Figura 4.29. Arquitectura del Lector de URL

A fin de modelar estas características se representa a cada agente con una arquitectura BDI, que, como se indicó en el capítulo 2 sección 2.1.3.1.5, abarca (de entre todas las distintas propuestas analizadas) de manera más general y completa, la definición de agente que se ha adoptado.

En la figura 4.28 se puede apreciar la arquitectura del agente Filtro de información. La interacción con el exterior se modela a través de un módulo Sensor, que se encarga de la comunicación hacia adentro, es decir monitorear el ambiente, ya sea por acciones del usuario como también por mensajes de otros agentes y transmitirlo a los Componentes reactivo y pro-activo; y de un módulo Efector, que se encarga de la comunicación hacia afuera, es decir comunicar al exterior los mensajes desde el Agente. A fin de modelar las características de reactividad, se cuenta con un módulo denominado Componente Reactivo, que incluye al Analizador Léxico encargado de examinar la entrada, un Clasificador encargado de producir una calificación de los artículos y producir la representación interna. Esta representación interna se comunica al Componente pro-activo, que modela la característica deliberativa del agente, y además al Efector de modo a que lo transmita a la sociedad multiagente. A fin de lograr la deliberación el Componente pro-activo posee un submódulo denominado Motor de Inferencia, que se comunica con los módulos Deseos, Creencias e Intenciones que modelan las nociones mentales del agente necesarias para completar la deliberación. Además existe una comunicación directa entre el módulo Sensor y el Componente Pro-activo, por medio de los mensajes provenientes de otros agentes, que pueden o no ser comunicados al usuario dependiendo de los planes e intenciones del Filtro de información.

En la figura 4.29, se presenta la arquitectura del agente Lector que al igual que el agente Filtro de información (y con el mismo propósito) posee los módulos Sensor y Efector. La diferencia principal

radica en que sólo posee el Componente Reactivo, que está compuesto de un Analizador Léxico, que se encarga de examinar la entrada.

4.3.2 PLATAFORMA DE IMPLEMENTACIÓN

En esta sección se detalla las herramientas de implementación utilizadas para la construcción del agente inteligente

Interfaz del Sistema

JBuilder Vx (Versión de Prueba)

Agentes:

- Filtro de información y Lector de URL:

j2sdk-1_4_2_03-windows-i586-p.exe,

j2re1_3_0-win.exe

Las herramientas utilizadas fueron seleccionadas sobre la base de que la plataforma destino de la aplicación es Win32 y con el componente adicional de ser un lenguaje que soporta el Paradigma Orientado a Objetos. De hecho Java es un lenguaje Orientado a Objetos.

4.3.3 ANALISIS COMPARATIVO

Como se había definido previamente, el presente trabajo de investigación se centra sobre las siguientes metodologías:

- Proceso Unificado Rational [KRUCH98, BRJ97]
- Técnica de Modelado de Agentes para Sistemas de agentes BDI [KINNY96]
- Metodología MAS-CommonKADS [IGLESI97]

Una descripción sintética de las mismas se puede encontrar en el capítulo 4, mientras que el lector interesado en mayores detalles, puede consultar las bibliografías correspondientes.

Creo definitivamente que las características del dominio de aplicación, además de influir en la arquitectura del agente, podrían tener también cierta influencia en la metodología seleccionada. Es por ello que este análisis se basa en un sistema que tienen características distintas y complementarias, para cubrir todas las características requeridas por la definición fuerte de Agente que se ha adoptado.

El prototipo FiltroInfo (Filtro de Información) descrito en este capítulo anteriormente, se desarrolló en su totalidad; siguiendo las fases de desarrollo de un producto software: conceptualización, análisis, diseño e implementación.

Como se puede notar, FiltroInfo posee todas las características presentes en la definición fuerte de agentes [WOOLDR95a]: autonomía, reactividad, sociabilidad, pro-actividad, nociones mentales, racionalidad y aprendizaje.

En los apéndices A y B, se pueden encontrar los diseños Orientado a Objetos y Orientados a Agentes respectivamente, correspondientes al prototipo FiltroInfo.

4.3.4 ANALISIS DE PROCESOS Y RESULTADOS

En esta sección se presenta un análisis y algunas consideraciones evaluativas de los procesos y de los resultados obtenidos al aplicar las distintas metodologías al desarrollo de sistemas basados en agentes.

Para realizar este análisis se ha considerado como esquema de trabajo la propuesta de un marco de evaluación para metodologías de diseño y construcción de aplicaciones hipermedia de [CERNUZ99, ORTIZ99]. El marco evaluativo consiste en la identificación de ciertos parámetros significativos y la verificación de su cumplimiento por parte de las metodologías en estudio, y se divide en aspectos generales del proceso de diseño y desarrollo de sistemas software y específicos de aplicaciones hipermedia. De esta propuesta, sólo se considera los parámetros generales.

Cabe además destacar que, siendo el objetivo general de la tesis la "*Las Metodologías Orientadas a Objetos con relación a las Metodologías Orientadas a Agentes facilitarán la construcción de agentes*", en el análisis de las metodologías estudiadas sólo dedicaré espacio a verificar hipótesis de extensión del Proceso Unificado Rational en los puntos que éste presente limitaciones.

En la sección 4.3.4.1, inicia con una visión global y luego, en la sección 4.3.4.2, se centra el análisis en cada etapa del proceso de desarrollo de un sistema basado en agentes.

4.3.4.1 Perspectiva Global o de Ciclo de Vida

Desde el nacimiento de una idea para un sistema software, hasta que ésta es implementada y entregada al cliente, y aún después de esto, el sistema software se desarrolla y evoluciona gradualmente en varias fases. Cada una de estas fases resulta en la construcción tanto de una parte del sistema como la de documentación asociada al mismo, tales como planes de verificación o manuales del usuario. Todo

este conjunto de etapas o fases por las que atraviesa el sistema software se conoce con el nombre de *ciclo de vida* del software. Existen varios enfoques o modelos de ciclo de vida (Waterfall [ROYCE70], de Transformación [VIENNE97], Evolucionario [GILB88], Espiral [BOEHM88], entre otros) y las distintas metodologías de desarrollo adoptan, de manera implícita o explícita, uno u otro de estos modelos.

De modo que se pueda manejar todo el proceso de desarrollo de un sistema software, es muy importante que la metodología empleada cubra todos los aspectos del ciclo de vida del sistema. Una de las limitaciones señaladas en [IGLESI97], es que las metodologías Orientadas a Agentes no siempre cumplen con este requerimiento, esto probablemente se debe a que las metodologías conocidas se encuentran aún en sus inicios, y a la falta de trabajos experimentales en los que se verifiquen los aporte de su aplicación en la construcción de sistemas reales.

A primera vista (sin precisar un análisis profundo) se puede notar (ver capítulo 2) que el Proceso Unificado Rational constituye una metodología iterativa e incremental y, al igual que ésta, MAS-CommonKADS propone un modelo en espiral dirigido por riesgos. Sin embargo la Técnica de Modelado de Agentes para Sistemas de agentes BDI, es mucho más reducida, más bien del tipo Waterfall [ROYCE70] sin cubrir además todas las fases del ciclo de vida.

Por ser el modelo en espiral más general -en efecto se trata de un meta modelo que puede contener en sus iteraciones Waterfalls y otros modelos- se lo adopta como marco referencial para el análisis. En particular se considera que cada iteración está compuesta de cinco etapas: Conceptualización, Análisis, Diseño, Implementación y Pruebas, y Operación y Mantenimiento. Al tomar esta decisión, que ayuda a lograr un análisis comparativo más específico, se está consciente de la imposibilidad de tratar cada etapa con rigidez ya que las fronteras entre ellas, sobre todo entre Conceptualización y Análisis, y Análisis y Diseño son bastante difusas.

Luego de la experiencia de desarrollo con las metodologías en estudio se destaca algunos puntos significativos:

El Proceso Unificado Rational

- Cubre todo el ciclo de vida de un sistema software en general.
- El nivel gradual de reducción de la abstracción, que adopta esta metodología, ayuda a la separación de conceptos y la generalidad durante el proceso.
- La modularidad es otro punto fuerte de la metodología, propio de toda metodología Orientada a Objetos.

- El traspaso de un modelo conceptual (arquitectura de módulos, ver figura A.23) a un modelo físico (asignación de módulos a procesadores, ver figura A.24) es sencillo y útil, ya que se finaliza con una vista conceptual y una correspondiente vista física del sistema.

La metodología MAS-CommonKADS

- Cubre los aspectos del ciclo de vida de un sistema basado en agentes.
- Las primeras etapas de Conceptualización y Análisis trabajan a un nivel elevado de abstracción permitiendo identificar los componentes del sistema agente, lo que se refina y profundiza, de manera sencilla y directa, en la etapa de Diseño.
- Sin embargo el traspaso de la arquitectura resultante a modelos de las etapas finales (Codificación, Integración, Entrega y Mantenimiento) es un tanto complicado e indirecto. Esto se puede deber, tal vez, a que no existe aún una plataforma de agentes definida para la implementación.
- Implementa un nivel de modularidad interesante ya que cada agente se modela mediante una definición independiente y relacionada a su vez con otros componentes modulares que representan ya sea sus tareas, objetivos o experiencias.
- Promueve la separación de conceptos con los diferentes modelos bien distinguibles, que propone para la etapa de Análisis.

La Técnica de Modelado de Agentes para Sistemas de agentes BDI

- No cubre todos los aspectos del ciclo de vida de un sistema basado en agentes.
- Abarca el Análisis y el Diseño a muy alto nivel, aunque particularmente interesante por el tratamiento más profundo de características que logra modelar, tal como las nociones mentales.
- La arquitectura resultante es muy abstracta para poder ser implementada directamente en cualquier plataforma de desarrollo, necesariamente debe ser refinada adoptando otra metodología. Consecuentemente todos los resultados obtenidos durante las primeras etapas, si bien constituyen una especificación relevante del sistema agente, ven limitados sus beneficios porque no se cuenta con guías consistentes y pasos directos para derivarlos de forma incremental en un código modular.
- La modularidad y separación de conceptos se implementan para las características de agentes, pero no se observa lo mismo con el resto de componentes (no agentes) del sistema.

Una limitación que comparten ambas Metodologías Orientadas a Agentes estudiadas, es la falta de herramientas CASE para el soporte del proceso de desarrollo de sistemas de agentes. Si bien estas

herramientas no constituyen un aporte conceptual a la metodología, ofrecen ventajas relevantes en el uso de las mismas. Facilitan una transición más natural entre las distintas etapas; aseguran un control de consistencia más preciso y completo; ahorran tiempo en el trabajo de documentación y automatizan ciertas tareas tediosas.

4.3.4.2 Análisis específico por etapa

En esta sección se presenta un análisis específico de las metodologías seleccionadas para la investigación considerando cada etapa del ciclo de vida de sistemas basados en agentes, así como se definió en la sección anterior.

4.3.4.2.1 *Conceptualización*

El propósito de esta etapa es comenzar a identificar los requerimientos de la aplicación, en términos de funcionalidad, desempeño, facilidad de uso, portabilidad, y otras características deseables del software. En esta etapa se especifican las necesidades del usuario.

Proceso Unificado Rational

Propone la construcción de diagramas de Casos de Uso (figura A.1 a A.3) y diagramas de Secuencia (figuras A.4 a A.10), que especifican la funcionalidad del sistema desde el punto de vista del usuario. Los diagramas de Secuencia, que se elaboran a partir de los diagramas de Casos de Uso, los cuales permiten identificar las entidades necesarias para completar la funcionalidad deseada del sistema. Estas entidades son objetos que determinarán posteriormente el vocabulario del dominio de aplicación. Por tanto podría ser un primer paso en la identificación de posibles agentes del sistema. En las figuras A4....A.10, se puede observar que los objetos Filtro, Lector, Analizador, Clasificador y Base de Datos, además de ser objetos activos, actúan como servidores del Usuario ya que toda acción de este actor es filtrada y realizada a través de estos objetos. Por ser una metodología Orientada a Objetos, evidentemente, carece de guías que, a partir de ciertas características del objeto, ayuden a catalogarlo como agente. No empero, permite la identificación y modelado de dichas características.

MAS-CommonKADS

La metodología propone para esta etapa, un análisis centrado en el usuario, para ello adopta la técnica de identificación de casos de uso de Jacobson [JACOBS97]. Incluye por cada Caso de Uso una notación gráfica (figura B.17). También, por cada Caso de Uso, incluye la descripción de la secuencia (figuras B.18 a B.20).

En esta etapa no se identifican aún las entidades de tipo agente. Como se puede notar el enfoque de esta metodología sigue los mismos pasos del Proceso Unificado Rational simplemente adoptando una notación diferente. Se considera que el aspecto más relevante es el énfasis puesto en la perspectiva del usuario facilitándose la identificación de las funcionalidades del sistema y su especificación a un nivel de abstracción elevado.

La Técnica de Modelado de agentes para sistemas de agentes BDI

No cubre una etapa de Conceptualización del sistema, asume que esta descripción ya existe modelada en algún formalismo.

Esta etapa debería finalizar con la identificación de los requerimientos del sistema que dependen de las necesidades del usuario. Como son:

- Usuarios del sistema,
- Funcionalidades del sistema, y
- Características no funcionales (desempeño, facilidad de uso, confiabilidad, etc.)

Se puede decir que las metodologías en estudio, que mejor cubren estos puntos son: el Proceso Unificado Rational y la metodología MAS-CommonKADS.

4.3.4.2.2 *Análisis*

El propósito de esta etapa es la determinación de los requisitos del sistema partiendo del enunciado del problema resultado de la etapa de conceptualización.

Proceso Unificado Rational

En esta etapa, propone la construcción de:

- Diagramas de Objetos: a partir de los objetos identificados en la etapa de conceptualización se construyen diagramas de objetos que dan una visión general del sistema, evidenciando todos los objetos componentes y las relaciones entre los mismos. En la figura A.11, es un diagrama general de objetos del sistema FiltroInfo. Se centra nuevamente el análisis en los objetos

Filtro, Lector, Analizador y Clasificador, ya que estos son críticos por poseer características de agentes. Se puede apreciar en la figura A.11, que la notación permite modelar:

- la relación entre las entidades de tipo agente con entidades convencionales (no agentes); por ejemplo, la relación Accede entre el Analizador y la Base de Datos.
- Diagrama de Colaboración: que da una vista dinámica centrada en el traspaso de mensajes entre los objetos. En la figura A.12, se tiene un diagrama que modela la comunicación entre objetos del sistema y el usuario. Se puede notar que los objetos Filtro, Lector, Analizador y Clasificador son objetos activos, lo que implica que tienen un hilo propio de control, con lo que se puede indicar la autonomía de estos objetos para generar los distintos mensajes y la concurrencia de ejecución.
- Diagrama de Clases: que da una vista de la estructura general del sistema y define el vocabulario del dominio de aplicación. En la figura A.13, se tiene el diagrama general de clases del sistema FiltroInfo. En éste se puede notar: que Filtro y Lector son especializaciones de una clase denominada Agente; que Filtro es una agregación de Analizador y Clasificador, el cual es especializado en agrupar y predecir; y que Lector es una agregación de Analizador-Léxico. En el capítulo 1, sección 1.4, se había especificado como puntos críticos para el paradigma Orientado a Objetos en el modelado de agentes:
    - El hecho de que los objetos responden a mensajes no estructurados propio de la definición de cada objeto, mientras que los agentes se comunican por medio de mensajes estructurados común a todos los agentes. Para lograr especificar esta característica de los agentes, se puede notar en la figura A.13, que se define una clase Lenguaje que implementa los métodos necesarios para comunicación de mensajes. Además, la clase Lenguaje es del tipo parametrizada con lo que se pretende modelar el hecho que el tipo o estructura del mensaje puede ser instanciado, de modo a que el lenguaje de comunicación de agentes puede ser cualquier lenguaje predefinido, por ejemplo KQML [FININ92]. Debido a que todos los agentes del sistema derivan de la clase Agente, que tiene como atributo a la clase Lenguaje, se consigue especificar que el lenguaje de comunicación es estructurado y común a todos;
    - Los agentes, a diferencia de los objetos, poseen además de datos y comportamiento, creencias, deseos e intenciones. Para lograr modelar esto, en la figura A.13, se puede notar que se identificaron las clases Creencias, Objetivos y Planes y que la clase Agente es una agregación de ellos. Estas clases tienen características especiales que se analizarán en la etapa de diseño.

En la figura A.13, se puede apreciar la clase Agente como agregación de Lenguaje, Creencias, Objetivos y Planes. Esta clase resulta de particular utilidad para modelar los agentes del

sistema FiltroInfo y ha sido creada con la intención de ser un patrón de definición de agente en varios dominios aplicativos, si bien no se puede asegurar su completitud para abarcar todo tipo de agente definible. Una generalización en este sentido requeriría un estudio más profundo que escapa al alcance del presente trabajo. Sin embargo se cree que podría ser de gran utilidad por lo especificado en los puntos anteriores.

Del análisis de los diagramas anteriores se puede notar que las clases Analizador, Clasificador, Filtro y Lector, poseen características de autonomía. Ya que la notación UML provee el concepto de actores como entidades autónomas y activas, se puede modelar a las entidades de tipo agente como clases con el estereotipo de actores. Esto implica, sin embargo, una extensión a la notación UML, ya que para ella, los actores son externos al sistema, mientras que las entidades agentes son internas al sistema. No obstante se considera que, eliminando esta restricción, serían de gran utilidad los diagramas de Casos de Uso y diagramas de Secuencia en los que los iniciadores son entidades internas del tipo agente, a fin de modelar la autonomía. Además la identificación de los Casos de Uso (ver figuras A.2 y A.3) iniciados por los agentes permite modelar los planes del agente a alto nivel, y los diagramas de Secuencia (ver figuras A.8 a A.10) permiten modelar en detalle el traspaso de mensajes entre agentes, y entre agentes y usuario.

MAS-CommonKADS

Propone la construcción de los modelos de:

- Tareas: muestra la descomposición funcional del sistema. Para ello se identifican las áreas funcionales del dominio de aplicación a partir del resultado de la etapa de conceptualización. En la figura B.21, se tienen el diagrama de descomposición de tareas, en los que se puede notar jerarquía y sincronización.
- Agentes: describe a los agentes que participan en la resolución del problema. Incluye una guía muy útil e interesante para la identificación de las entidades del sistema que pueden ser modeladas como agentes. Luego de esta identificación se construyen plantillas de agentes preliminares.

Luego de la descripción de los agentes del sistema, se especifica la asignación de tareas a agentes (ver apéndice B, sección B.2.4.2) y la descripción por cada agente de sus objetivos (ver apéndice B, sección B.2.4.3). No permite, sin embargo, modelar objetivos subjetivos o difusos, ni comportamiento evolutivo, necesarios para especificar las características de pro-actividad del agente.

- Experiencia: modela el conocimiento de cada agente necesario para cumplir con los objetivos y tareas que posee. En el apéndice B, sección B.2.5, se puede encontrar la identificación y descripción del esquema de conocimiento del sistema FiltroInfo. Con este modelo se pretende especificar las creencias del agente, sin embargo no es posible modelar creencias difusas y dinámicas, necesarias para las características de aprendizaje del agente.
- Comunicación: se encarga del modelado de las relaciones hombre-máquina. La metodología contempla esta característica pero no ofrece un modelo conceptual para especificarla.
- Coordinación: su principal objetivo es modelar la interacción agente-agente. Para ello se identifican y describen las conversaciones entre agentes, y a partir de ellas las intervenciones y los servicios derivados (ver apéndice B, sección B.2.6).
- Organización: modela las relaciones estáticas entre los agentes del sistema. En el apéndice B, sección B.2.8, se presenta la estructura organizativa de los agentes y la relación con el resto de los objetos del sistema. Se lo considera un modelo útil ya que provee una vista global del sistema completo.

Técnica de Modelado de Agentes para sistemas de agentes BDI

Esta metodología, en cambio, define en la etapa de análisis la construcción de la vista externa del sistema. Para ello propone los siguientes modelos:

- de Agentes: que describe la relación jerárquica entre las diferentes clases de agentes. Como primer paso se identifican los roles del dominio de aplicación que determinan a los agentes del sistema. A partir de estos roles se construye un Modelo de Clases de Agentes (ver figura B.1) y un Modelo de Instancias de Agentes (ver figura B.2). Para la metodología, toda entidad del sistema debe ser necesariamente un agente, lo que lleva a identificar agentes que paradójicamente, según la definición que soporta la metodología, no son agentes. Este problema se hace aún más evidente en la etapa de diseño.
- de Interacción: refina el modelo anterior identificando las responsabilidades y servicios de cada agente. En el apéndice B, sección B.1.2, se presenta la descripción de la interacción del sistema FiltroInfo. Se considera que la serie de pasos definidos en la construcción de este modelo es muy interesante debido a que permite identificar en detalle los mensajes y el orden de los mismos para la interacción tanto entre agentes como entre agentes y usuario.

Esta etapa debería finalizar con la identificación y especificación de los requerimientos del software que dependen de la identificación de los requerimientos del sistema (etapa de conceptualización). Para el caso de sistemas de agentes, interesa identificar:

1. Entidades estáticas del dominio de aplicación (objetos convencionales)
2. Entidades autónomas y dinámicas del dominio de aplicación (agentes)
3. Relaciones entre las entidades constituyentes
4. Interacción entre las entidades constituyentes (objeto-agente, objeto-objeto, agente-agente)

El enfoque de la Técnica de Modelado de Agentes para sistemas de agentes BDI para los puntos 2 y 4, es muy interesante y a mi parecer bastante completo, pero la metodología no abarca de manera satisfactoria el resto de los puntos. El Proceso Unificado Rational abarca de manera completa y bastante satisfactoria los puntos 1, 3 y 4; mientras que para el punto 2 le faltarían guías de identificación que ayuden a derivar entidades del tipo agente. MAS-CommonKADS, por su parte, trata de manera interesante los puntos 2, 3 y parcialmente el 4, ya que en este último no ha definido aún el modelo de especificación para la interacción objeto-agente. El punto 1, se trata de manera interna y muy superficial al abarcar el punto 2.

4.3.4.2.3 *Diseño*

El objetivo de esta etapa es lograr una descripción de la arquitectura del sistema en términos de módulos y relación entre ellos.

Proceso Unificado Rational

Propone la construcción de los siguientes diagramas:

- Diagrama de clases: se refina el diagrama correspondiente de la etapa de anterior completando la descripción de cada clase del sistema. Como se lo especifica en la sección anterior, en el diagrama general de clases (figura A.13), se incluyeron las clases Creencias, Objetivos y Planes, que poseen características dinámicas y evolutivas no definidas en tiempo de especificación. La notación UML, sin embargo, no contempla la especificación de este tipo de características. No obstante, se necesita especificar ya que constituyen elementos indispensables y críticos para la definición de agentes adoptada [WOOLDR95a]. Es por ello que se amplia la descripción de estas clases utilizando el lenguaje formal dyOSL [SAAKE95]. En las figuras A.16 y A.17, se presenta la descripción de las clases Creencias y Objetivos en dyOSL. Fue posible la especificación del comportamiento evolutivo en tiempo de ejecución (importante para el aprendizaje del agente). El comportamiento de la clase Planes, fue modelodo mediante una serie de diagramas de transición de estados (figuras A.18 a A.21), con lo que se pudo captar la secuencia necesaria para cumplir un objetivo dado.

- Diagrama de Transición de Estados: modela la estructura dinámica del sistema orientado a eventos. Como ya se especifico en la sección anterior, este diagrama fue útil en el modelado de los planes de un agente. Además en las figuras A.14 y A.15 gracias a estos diagramas se pudieron especificar los distintos estados en los que pueden encontrarse los agentes Filtro y Lector respectivamente.
- Diagrama de Módulos: modela la arquitectura global del sistema a partir de módulos y relaciones entre estos. Luego de haber identificado la totalidad de clases, que definen al dominio de aplicación así como las relaciones entre ellas, se las agrupan en módulos, de acuerdo a la dependencia lógica entre las mismas. En la figura A.22, se tiene el diagrama de módulos del sistema FiltroInfo.

MAS-CommonKADS

La metodología propone para esta etapa la construcción de un Modelo de Diseño. Este modelo incluye diseño del agente (aplicación), diseño de la red (arquitectura) y diseño de la plataforma (ver apéndice B, sección B.2.9). No obstante, se considera la definición del modelo un tanto débil, debido a que la descripción de cada diseño es bastante superficial. Además, la metodología tampoco provee pautas claras acerca de cómo traducir las especificaciones previas a una arquitectura particular que modele el sistema agente. Por ejemplo, al especificar la arquitectura de un agente simplemente se enuncia el tipo (reactivo, deliberativo o híbrido) y, como se comprobó en el capítulo 2,, existen, a su vez, por cada arquitectura, varios enfoques que se podrían adoptar.

Técnica de Modelado de Agentes para Sistemas de Agentes BDI

En la etapa de diseño, define la vista interna del sistema. Incluye:

- Modelo de Creencias: describe la información acerca del ambiente, el estado interno que cada agente puede poseer, y las acciones que puede realizar. En el apéndice B, sección B.1.3, se puede encontrar la definición del modelo de Creencias del sistema FiltroInfo. Este modelo no permite especificar agregación, modificación o eliminación de creencias, necesarios para modelar el aprendizaje del agente. Tampoco es posible modelar incertidumbre ya que en la especificación se utilizan funciones y predicados de la lógica de primer orden.
- Modelo de Objetivos: modela las intenciones del agente (ver figura B.12). Contempla sólo tres tipos de objetivos: lograr, preguntar y verificar. No permite objetivos subjetivos ni tampoco soporta modelado de comportamiento evolutivo de estos, necesario para captar los requerimientos de aprendizaje del agente.

- Modelo de Planes: especifica el plan a seguir por el agente de modo a lograr un objetivo determinado (ver apéndice B, sección B.1.5). Se considera al modelo completo e intuitivo para la especificación de las intenciones del agente.

Esta etapa debería finalizar con la identificación y especificación de la arquitectura del sistema. Esto incluye:

1. Identificación y especificación interna de todos los módulos del sistema, e
2. Identificación y especificación de relaciones entre los módulos.

Definitivamente de lo expuesto en los párrafos anteriores, queda claro que el Proceso Unificado Rational, de entre las estudiadas es la metodología que abarca de manera más satisfactoria ambos puntos.

4.3.4.2.4 *Implementación y Verificación*

El resultado de esta etapa es una colección de módulos implementados y verificados.

Proceso Unificado Rational

Antes de iniciar la codificación propiamente dicha, el Proceso Unificado Rational, propone la construcción del diagrama de Procesadores, que da una vista física de la arquitectura del sistema en cuanto a la asignación de módulos a procesadores. En este caso solo se presenta un único procesador.

Para la codificación de los distintos módulos se trabaja basándose en la estructura dinámica de la aplicación definida en los diagramas de Secuencia y de Colaboración del sistema. Se selecciona un lenguaje de programación que puede o no ser Orientado a Objetos. Una vez finalizada la construcción de los módulos se procede al testeo de los mismos para lo cual se utilizan todos los diagramas definidos en UML.

MAS-CommonKADS

La metodología, no define pasos claros para esta etapa. Lo establece como una cuestión abierta. Para la verificación propone comprobar parcialmente la corrección de la conducta global del sistema utilizando los escenarios típicos que tratan los posibles conflictos y los métodos de resolución de los mismos.

Técnica de Modelado de Agentes para Sistemas de Agentes BDI

No cubre esta etapa del ciclo de vida.

En este capítulo, sección 4.1.3, se puede encontrar el detalle de los lenguajes seleccionados para la implementación de los distintos módulos del sistema FiltroInfo, junto con la justificación de la elección. Todo el proceso de desarrollo del sistema FiltroInfo fue implementado en el lenguaje Java.

4.3.4.2.5 *Operación y Mantenimiento*

Es la etapa en la que se entrega el sistema software al usuario y se corrigen posibles errores del sistema o se agregan funcionalidades al mismo.

En esta etapa el Proceso Unificado Rational propone la entrega del producto, validación por parte del usuario y mantenimiento, siguiendo el mismo proceso en espiral dirigido por riesgos propuesto para todo el proceso de desarrollo.

Tanto la metodología MAS-CommonKADS como la Técnica de Modelado de Agentes para Sistemas de Agentes BDI, no cubren esta etapa del ciclo de vida.

4.3.5 RESUMEN COMPARATIVO

En las secciones anteriores se ha centrado el análisis en el proceso de construcción de agentes. Sin embargo no se puede dejar de considerar que algunos de los principales objetivos del proceso son:

- Incrementar la eficiencia,
- Asegurar una mayor calidad del producto,
- Proveer una adecuada documentación,
- Facilitar la reusabilidad y mantenimiento de los componentes del sistema.

Los resultados del proceso se ven reflejados en el cumplimiento de los requerimientos del producto que en este caso son sistemas agentes. Para el efecto en la tabla 4.2 se analiza, según las distintas

características de los agentes, de qué manera cada metodología estudiada abarca o no la especificación de las mismas.

Característica	**Proceso Unificado Rational**	**MAS-CommonKADS**	**Técnica de Modelado de Agentes para Sistemas de Agentes BDI.**
Autonomía	Cumple	Cumple	Cumple
Reactividad	Cumple	Cumple	Cumple
Pro-actividad	Cumple	Cumple	Cumple
Nociones Mentales	~	~	Cumple
Racionalidad	No cumple	No cumple	No cumple
Cooperación con otros agentes	Cumple	Cumple	Cumple
Cooperación con entidades no agentes	Cumple	Aún no esta definido el modelo	No cumple
Comportamiento evolutivo	Cumple	No cumple	No cumple
Tiempo de Vida	Cumple	No cumple	No cumple
Aprendizaje	No cumple	No cumple	No cumple

Tabla 4.2: Análisis Comparativo

La mayoría de los elementos de la tabla 4.2 son auto-explicativos y la única excepción se refiere al símbolo ~ que indica que la metodología cubre parcialmente la especificación de la característica analizada. Esto ocurre en los siguientes casos:

Nociones Mentales:

- Con el Proceso Unificado Rational se puede especificar que el agente posee Objetivos, Creencias y Planes. Con el lenguaje dyOSL se puede modelar el comportamiento evolutivo de los mismos. Pero no es posible explicitar cuáles son los Objetivos y Creencias concretos. Sin embargo para el caso de los Planes, como se vio en la sección 4.3.4.1, los diagramas de transición de estados son muy adecuados ya que modelan la secuencia de pasos según los

eventos, a fin de lograr los mismos. Es por estas razones que se considera que la característica de Nociones Mentales no es cubierta en su totalidad.

- Con MAS-CommonKADS se puede especificar que el agente posee Objetivos, requiere cierta capacidad para realizar tareas y además posee Experiencias (Creencias o Conocimiento). No abarca, sin embargo, los Planes del agente.

4.3.6 PROCESO UNIFICADO RATIONAL EXTENDIDO

No se puede considerar los resultados obtenidos definitivos, ni siquiera completos para todo tipo de agente. Pero, considerando las limitaciones que presentan también las metodologías específicas de agentes, se puede concluir con cierto grado de seguridad que el paradigma Orientado a Objetos es una herramienta poderosa para el diseño de agentes. Al respecto considero que el Proceso Unificado Rational es una buena metodología, si bien es necesario extenderla con formalismos adecuados para cubrir el modelado de las características de sistemas basados en agentes.

Algunas de estas extensiones ya fueron aplicadas en el presente trabajo de investigación y probaron ser adecuadas:

- Actores como entidades internas del sistema para modelar agentes.
- Lenguaje de especificación de objetos dinámicos dyOSL.

Para las características de:

- Nociones Mentales, Racionalidad y Aprendizaje: podría ser conveniente adoptar algún lenguaje de especificación del área de Ingeniería del Conocimiento, que abarque el modelado de las mismas, e incluir la definición de estas como una etapa más en la fase de diseño, complementando así la descripción de las clases que representan a los agentes.
- Lenguaje de comunicación estructurado y común a todos los agentes: podría ser interesante analizar propuestas del tipo CORBA del grupo Object Management Group y la arquitectura OLE 2/ActiveX de Microsoft. En estas el propósito general es permitir a las aplicaciones acceder a los objetos y servicios de cada una de la manera más conveniente, tanto local como en forma remota. También han establecido énfasis particular en la interoperabilidad, que se aplica al nivel de lenguaje. Gracias a esto, las aplicaciones escritas en un lenguaje pueden acceder a objetos de otra aplicación escritos en otro lenguaje. Esta interacción es posible a través de un lenguaje intermedio llamado IDL (Interface Definition Language – Lenguaje de Definición de Interfaz).

4.3.7 PRUEBAS DEL PROTOTIPO

La aplicación FiltroInfo desarrollada provee la funcionalidad básica de un filtro de información de propósito general. EL Agente es provisto para recoger información desde páginas Web del Internet. El Agente Lector de Paginas Web lee los datos desde un URL (*Uniform Resource Locator*) de una página Web específica y los descarga al PC. Después cada artículo o página Web es calificado de acuerdo a una lista de palabras claves y/o por medio del uso de dos redes neuronales a través del Agente Filtro.

Las paginas web son cargadas, ordenadas y agrupadas de acuerdo a su calificación y nivel de interés para el usuario:

- **Para el** Filtro Palabras Claves**, la calificación es simplemente la suma de todas las palabras claves (por default), y se le califica como interesante por el mayor número de palabras claves que contiene.**
- **Para el** Filtro Cluster**, esta calificación es el promedio de las calificaciones de las palabras claves de todos los artículos que están dentro de este grupo, mientras mayor es el promedio del grupo es más interesante.**
- **Para el** Filtro Feedback, **se crea un modelo de predicción, mediante el cual califica la similitud de la pagina que se ingresa con respecto a un perfil de usuario ya cargado y analizado**

Antes de realizar la clasificación se debe añadir las páginas web de interés para poder entrenar a las redes neuronales tanto para el filtro Cluster como para el filtro Feedback de acuerdo con las palabras claves que contienen cada página añadida.

En esta fase se desarrolla una experiencia de evaluación heurística [NIELSE94] del prototipo FiltroInfo. Para ello se lleva a cabo una experiencia en dos etapas, con un grupo de 10 alumnos de la Facultad de Ingeniería en Sistemas e Informática de la ESPE. En la primera etapa se dirigió una prueba inicial en el que el agente FiltroInfo adquirió experiencia a partir de la asistencia que otorgó a cada alumno. En la segunda etapa se repitió el testeo con el mismo grupo
de alumnos, pero con una base de conocimiento más amplia del agente construida a partir de la experiencia obtenida en la etapa anterior. Esta experiencia arrojó resultados interesantes, respecto a las alternativas de clasificación de los documentos (Palabras Claves, Clustering, Feedback).

4.3.8 Resultados

Para realizar las pruebas del prototipo se seleccionó en primer lugar las palabras claves que van a intervenir en la clasificación de páginas web:

No.	**Palabras Claves**

1	agente
2	inteligente
3	red
4	neuronal

Figura 4.30: Palabras Claves

Las palabras claves escogidas son inherentes al tema, Agentes Inteligentes, y son las que se quiere encontrar en páginas webs.

Las páginas web a ser visitadas por FiltroInfo son:

Ord.	Página Web	Descripción
1	http://www.acm.org/crossroads/espanol/xrds5-4/dumbagents.html	¿De dónde vienen los agentes inteligentes?
2	http://www.redcientifica.com/doc/doc199903310001.html	Agentes Autónomos Inteligentes
3	http://club.telepolis.com/ohcop/neuretol.html	Alarmas en robótica neuroetológica
4	http://www.sia.eui.upm.es/docent/ai.html	La asignatura de Agentes Inteligentes de la UPM
5	http://seneca.uab.es/prehistoria/Barcelo/TyTAI_Intro.html	Introducción a la Inteligencia Artificial en Arqueología
6	http://geneura.ugr.es/~jmerelo/atalaya/agentes.htm	Agentes inteligentes

Figura 4.31: Páginas Web seleccionadas

Las Paginas Web seleccionadas están relacionadas con el tema.
Para la clasificación de páginas web se procedió a cargar las páginas web en la aplicación FiltroInfo, obteniéndose los siguientes resultados:

Tres Páginas Web

Al inicio se cargó las dos primeras páginas, con las cuales se obtuvo los siguientes resultados en los diferentes tipos de filtros (Palabras Claves, Clustering, Feedback).

Palabras Claves

Ord.	Página Web	Calificación	Nivel

1	http://www.acm.org/crossroads/espanol/xrds5-4/dumbagents.html	15	Interesante
2	http://www.redcientifica.com/doc/doc199903310001.html	24	Interesante

TABLA 4.3: Clasificación Inicial de dos páginas web (Filtro: Palabras Claves)

Interpretación

Las páginas web fueron clasificadas por medio del Filtro "Palabras Claves"; es decir cada página web que es cargada dentro de FiltroInfo busca el número de veces que se repite cada palabra clave, la cual al momento que aparece es contada y sumada, es decir que para la primera dirección web existe 15 palabras claves, en cambio en la segunda dirección web solo aparecen 24 palabras claves dentro del documento, pudiendo ser estas palabras claves repetidas varias veces en la página.

Luego se procedió a entrenar la red y se obtiene la clasificación correspondiente a Cluster y Feedback.

Cluster

Ord.	Página Web	Calificación	Nivel
1	http://www.redcientifica.com/doc/doc199903310001.html	24.0	Interesante
2	http://www.acm.org/crossroads/espanol/xrds5-4/dumbagents.html	15.0	Interesante

TABLA 4.4: Clasificación de tres páginas web (Filtro: Cluster)

Interpretación

Las páginas web fueron clasificadas por medio del Filtro "Cluster"; en este caso se forman 2 grupos, debido a que no todas las palabras han sido similares para conformar un solo grupo.

Feedback

Ord.	Página Web	Calificación	Nivel
1	http://www.acm.org/crossroads/espanol/xrds5-4/dumbagents.html	0.919101573783	Interesante
2	http://www.redcientifica.com/doc/doc199903310001.html	0.918415087181	Interesante

TABLA 4.5: Clasificación de páginas web (Filtro: Feedback)

Interpretación

Las páginas web fueron clasificadas por medio del Filtro “Feedback”; en este caso se toma en cuenta, no la cantidad de palabras claves repetidas, sino el número de palabras claves diferentes, es por eso que se invierte el orden de las páginas web en la clasificación.

Luego se ingresa en la aplicación la tercera pagina web resultando la siguiente clasificación:

Palabras Claves

Ord.	Página Web	Calificación	Nivel
1	http://www.redcientifica.com/doc/doc199903310001.html	25	Interesante
2	http://www.acm.org/crossroads/espanol/xrds5-4/dumbagents.html	15	Interesante
3	http://club.telepolis.com/ohcop/neuretol.html	9	Interesante

TABLA 4.6: Clasificación de tres páginas web (Filtro: Palabras Claves)

Interpretación

Las páginas web fueron clasificadas por medio del Filtro “Palabras Claves”; en este caso la página web ingresada tiene 9 palabras claves en su contenido, pudiendo ser éstas repetidas.

Cluster

Ord.	Página Web	Calificación	Nivel
1	http://www.redcientifica.com/doc/doc199903310001.html	24.0	Interesante
2	http://www.acm.org/crossroads/espanol/xrds5-4/dumbagents.html	12.0	Interesante
3	http://club.telepolis.com/ohcop/neuretol.html	12.0	Interesante

Tabla 4.7: Clasificación de tres páginas web (Filtro: Cluster)

Interpretación

Las páginas web fueron clasificadas por medio del Filtro “Cluster”; en este caso la página web ingresada tiene 9 palabras claves en su contenido y tiene la suficiente cantidad de palabras claves para formar parte de uno de los grupos anteriores.

Feedback

Ord.	Página Web	Calificación	Nivel
1	http://www.acm.org/crossroads/espanol/xrds5-4/dumbagents.html	0.919101573783	Interesante
2	http://www.redcientifica.com/doc/doc199903310001.html	0.918415087181	Interesante
3	http://club.telepolis.com/ohcop/neuretol.html	0.015989699871	Interesante

TABLA 4.8: Clasificación de páginas web (Filtro: Feedback)

Interpretación

Las páginas web fueron clasificadas por medio del Filtro "Feedback"; en este caso se toma en cuenta, no la cantidad de palabras claves repetidas, sino el número de palabras claves diferentes, es por eso que se invierte el orden de las páginas web en la clasificación.

De las interpretaciones se puede decir que la página de redcientifica tiene el mayor número de palabras claves, aunque éstas sean repetidas; una vez entrenadas las redes se forman en el método cluster dos grupos, de los cuales el que tiene mayor calificación es el de la página redcientifica; en cambio, por el método de propagación hacia atrás la página acm.org es la que contiene el mayor número de palabras claves y que se ajustan al perfil de usuario formado por las dos paginas anteriores: acm.org y redcientifica.

De lo anterior se puede deducir que la página que recomienda el sistema a acceder es acm.org, debido a que contiene el mayor número de palabras claves y que se ajusta al perfil del usuario.

Seis Páginas Web

Al inicio se cargó las cuatro primeras páginas, con las cuales se obtuvo los siguientes resultados en los diferentes tipos de filtros (Palabras Claves, Clustering, Feedback).

Palabras Claves

Ord.	Página Web	Calificación	Nivel
1	http://www.redcientifica.com/doc/doc199903310001.html	24	Interesante
2	http://www.acm.org/crossroads/espanol/xrds5-4/dumbagents.html	15	Interesante

3	http://club.telepolis.com/ohcop/neuretol.html	11	Interesante
4	http://www.sia.eui.upm.es/docent/ai.html	9	Interesante

TABLA 4.9: Clasificación Inicial de cuatro páginas web (Filtro: Palabras Claves)

Interpretación

Las páginas web fueron clasificadas por medio del Filtro "Palabras Claves"; es decir cada página web que es cargada dentro de FiltroInfo busca el número de veces que cada palabra clave aparece, la cual es contada y sumada, es decir que para la primera dirección web existe 24, en cambio en la segunda aparecen 15, en la tercera 11 y en la cuarta 9 palabras claves dentro de las páginas cargadas, pudiendo ser estas palabras claves repetidas varias veces en la página.

Luego se procedió a entrenar la red y se obtiene la clasificación correspondiente a Cluster y Feedback.

Cluster

Ord.	Página Web	Calificación	Nivel
1	http://www.acm.org/crossroads/espanol/xrds5-4/dumbagents.html	19.5	Interesante
2	http://www.redcientifica.com/doc/doc199903310001.html	19.5	Interesante
3	http://club.telepolis.com/ohcop/neuretol.html	10.0	Interesante
4	http://www.sia.eui.upm.es/docent/ai.html	10.0	Interesante

TABLA 4.10: Clasificación de cuatro páginas web (Filtro: Cluster)

Interpretación

Las páginas web fueron clasificadas por medio del Filtro "Cluster"; en este caso se forman 2 grupos, debido a que las dos primeras forman un grupo y las ultimas forman el otro grupo.

Feedback

Ord.	Página Web	Calificación	Nivel
1	http://www.acm.org/crossroads/espanol/xrds5-4/dumbagents.html	0953927099781	Interesante
2	http://www.redcientifica.com/doc/doc199903310001.html	0950137897180	Interesante
3	http://club.telepolis.com/ohcop/neuretol.html	0949226268261	Interesante
4	http://www.sia.eui.upm.es/docent/ai.html	0946916472248	Interesante

TABLA 4.11: Clasificación de cuatro páginas web (Filtro: Feedback)

Interpretación

Las páginas web fueron clasificadas por medio del Filtro “Feedback”; en este caso se toma en cuenta, no la cantidad de palabras claves repetidas, sino el número de palabras claves diferentes.

Luego se ingresa en la aplicación otras dos páginas web resultando la siguiente clasificación:

Palabras Claves

Ord.	Página Web	Calificación	Nivel
1	http://www.redcientifica.com/doc/doc199903310001.html	24	Interesante
2	http://www.acm.org/crossroads/espanol/xrds5-4/dumbagents.html	15	Interesante
3	http://seneca.uab.es/prehistoria/Barcelo/TyTAI_Intro.html	13	Interesante
4	http://club.telepolis.com/ohcop/neuretol.html	11	Interesante
5	http://www.sia.eui.upm.es/docent/ai.html	9	Interesante
6	http://geneura.ugr.es/~jmerelo/atalaya/agentes.htm	3	Normal

TABLA 4.12: Clasificación de seis páginas web (Filtro: Palabras Claves)

Interpretación

Las páginas web fueron clasificadas por medio del Filtro “Palabras Claves”; en este caso la páginas fueron ordenadas por la cantidad de palabras claves en su contenido, las mismas que pueden ser repetidas.

Cluster

Ord.	Página Web	Calificación	Nivel
1	http://www.acm.org/crossroads/espanol/xrds5-4/dumbagents.html	19.5	Interesante
2	http://www.redcientifica.com/doc/doc199903310001.html	19.5	Interesante
3	http://club.telepolis.com/ohcop/neuretol.html	10.0	Interesante
4	http://www.sia.eui.upm.es/docent/ai.html	10.0	Interesante

5	http://seneca.uab.es/prehistoria/Barcelo/TyTAI_Intro.html	8.0	Interesante
6	http://geneura.ugr.es/~jmerelo/atalaya/agentes.htm	8.0	Normal

TABLA 4.13: Clasificación de seis páginas web (Filtro: Cluster)

Interpretación

Las páginas web fueron clasificadas por medio del Filtro "Cluster"; en este caso las páginas web forman tres grupos, los mismos que se conforman de acuerdo con la cantidad de palabras similares que tiene y el perfil del usuario.

Feedback

Ord.	Página Web	Calificación	Nivel
1	http://www.redcientifica.com/doc/doc199903310001.html	0.934319807175	Interesante
2	http://www.acm.org/crossroads/espanol/xrds5-4/dumbagents.html	0.933515651810	Interesante
3	http://www.sia.eui.upm.es/docent/ai.html	0.922148005861	Interesante
4	http://geneura.ugr.es/~jmerelo/atalaya/agentes.htm	0.906105776780	Normal
5	http://club.telepolis.com/ohcop/neuretol.html	0-904443858365	Interesante
6	http://www.sia.eui.upm.es/docent/ai.html	0.814797526835	Interesante

TABLA 4.14: Clasificación de seis páginas web (Filtro: Feedback)

Interpretación

Las páginas web fueron clasificadas por medio del Filtro "Feedback"; en este caso de acuerdo al perfil de usuario y el número de palabras claves diferentes en las páginas se obtiene que acm.org es la página con el mayor número de palabras claves.

De las interpretaciones se puede decir que la página redcientifica tiene el mayor número de palabras claves, aunque éstas sean repetidas; una vez entrenadas las redes se forman en el método cluster tres grupos, de los cuales el que tiene mayor calificación son las páginas acm.org y redcientifica; en cambio, por el método de propagación hacia atrás la página redcientifica es la que contiene el mayor número de palabras claves y que se ajustan al perfil de usuario formado por las cuatro páginas anteriores: acm.org, redcientifica, club.telepolis y sia.eui.upm.es

De lo anterior se puede deducir que la página que recomienda el sistema a acceder es redcientifica, debido a que contiene el mayor número de palabras claves y que se ajusta al perfil del usuario.

4.3.9 DISCUSIÓN DE RESULTADOS

Quizá el principal problema en esta aplicación es la relación entre el código de la aplicación y el agente inteligente. Uno puede fácilmente imaginar una implementación en la cual el Agente Filtro no exista y estas funciones son realizadas por el programa principal. Sin embargo, se pueda reutilizar el Agente Filtro en una composición de agentes o sistema multiagente. Mientras el Agente Filtro no es autónomo como el Agente Lector de URL´s, éste en cambio entrena el modelo de red neuronal asincrónicamente. Como un pequeño trabajo adicional, el Agente Filtro debería ser modificado para construir modelos de redes neuronales para cualquier seteo de datos.
En esta aplicación se desarrolla una sencilla exploración de datos. Otros Agentes pueden acceder a una base de datos u otras fuentes de datos (la aplicación FiltroInfo accesa a fuentes de páginas Web) mediante un sistema autónomo de exploración de datos que puede ser construido.

Otro problema de diseño es la forma en la cual la búsqueda de palabras claves es realizada en el Agente Filtro, solo las palabras claves completas son contadas. Hay algoritmos para realizar semejanzas parciales de modo que palabras en singular y plural son contadas. Las palabras claves completas, en el texto, seguidas de signos de puntuación también son tomadas en cuenta, para ser contadas. Obviamente, la mejor información que se puede encontrar es la que se puede filtrar más cuidadosamente.

Para usar el filtro Cluster o el filtro Feedback, primero debe ser guardado el perfil de datos con un conjunto razonable de artículos.

Para usar el filtro Cluster, este debe ser seleccionado como el tipo de filtro, la lista de páginas web será renovado con las páginas web del primer grupo con un puntaje más alto, el segundo mejor grupo que le sigue en puntajes, y así sucesivamente.
El último filtro es Feedback, éste usa un modelo de red neuronal para determinar la calificación de páginas web. Crea un modelo de predicción back propagation usando los registros de perfil de páginas web como entrada y el usuario clasifica el puntaje mediante el valor de salida final. Si ninguno de los seteos de retroalimentación por defecto fue sobrescrito antes de que las páginas web fueran añadidos al perfil, el modelo predecirá una calificación desde 0.0 a 1.0, donde un artículo que tiene más de cinco palabras claves tendrá un valor de 1. Si valores diferentes fueran dados a páginas web seleccionadas por retroalimentación, usando los cinco niveles definidos en el menú Feedback, los resultados serían ligeramente diferentes. Por ejemplo, si una página web que contiene la palabra “agente” fue

clasificado como más interesante que uno que contiene la palabra “neuronal”, éste tendría calificación más alta en el filtro Feedback (aunque cada uno tuvo una calificación similar).

CONCLUSIONES

Una vez realizada la investigación formulada en esta tesis se puede concluir que:

La metodología orientada a objetos puede utilizarse en el análisis, diseño e implementación de agentes de software, ya que se ha demostrado que puede servir de base para la implementación de una metodología orientada a agentes.

Luego de haber realizado un análisis individual y comparativo entre las metodologías orientadas a objetos y las orientadas a agentes seleccionadas, en cuanto a sus etapas de desarrollo, se puede colegir que la metodología más factible en la construcción de agentes de software es la denominada “Proceso Unificado Rational”, por cuanto es capaz de modelar: 1) Autonomía, 2) reactividad, 3) proactividad, 4) cooperación con otros agentes, 5) cooperación con entidades no agentes, 6) comportamiento evolutivo y 7) su tiempo de vida.

En cuanto a la aplicación, el objetivo de ésta es ayudar al usuario a tratar de mostrar como una aplicación basada en agentes puede ser usada para ordenar inteligentemente páginas web basadas en su contenido, por cuanto la aplicación ***FiltroInfo*** provee tres tipos de filtrado para páginas web: 1) el filtro **Palabras Claves** contabiliza el número de palabras claves encontradas, 2) el filtro **Cluster** agrupa paginas similares y 3) el filtro **Feedback** usa un modelo predictivo basado en niveles de interés.

RECOMENDACIONES

Espero que este estudio y sus resultados hayan contribuido a una línea de investigación y puedan servir como base para trabajos futuros en el área.

Se podría experimentar extensiones a la Metodología del Proceso Unificado Rational a fin de cubrir mejor el modelado de las siete características de sistemas basados en agentes.

Se podría ampliar el sistema para que permita tener otro tipo de agente como un **Agente Lector de Noticias,** que puede conectarse a un servidor de noticias específico, solicitar artículos, descargarlos, clasificarlos y formar un perfil de usuario y así poder filtrar únicamente la información de temas de interés.

En este sentido podría ser interesante enfocar un agente para filtrar otros tipo de información, utilizando no solo simples palabras claves sino el singular o plural de las palabras, verbos y sus conjugaciones pasadas, presentes y futuras, así como frases, aumentando la capacidad del sistema en cuanto a conocimiento de la información filtrada. Además se puede aplicar otras técnicas que permitan identificar el interés del usuario (o carencia de el) en documentos específicos.

Se podrían desarrollar aplicaciones de agentes en varias áreas, entre las cuales estarían áreas particularmente significativas como la Informática Educativa sobre la cual no existen muchos trabajos.